

# **THE CIP NETWORKS LIBRARY**

## **Volume 3**

### **DeviceNet Adaptation of CIP**

---

Edition 1.10

November 2010

The CIP Networks Library  
Volume 3: DeviceNet Adaptation of CIP

Publication Number: PUB00003

Copyright © 1994 through 2010 ODVA, Inc. All rights reserved. For permissions to reproduce excerpts of this material, with appropriate attribution to the author(s), please contact ODVA at:

ODVA, Inc.  
4220 Varsity Drive, Suite A, Ann Arbor, MI 48108-5006 USA  
TEL 1-734-975-8840  
FAX 1-734-922-0027  
EMAIL [odva@odva.org](mailto:odva@odva.org)  
WEB [www.odva.org](http://www.odva.org)

#### Warranty Disclaimer Statement

The right to make, use, or sell product or system implementations based upon the Common Industrial Protocol (CIP) is granted only under separate license pursuant to a Terms of Usage Agreement or other agreement. The ODVA Terms of Usage Agreement is available, at standard charges, over the Internet at [www.odva.org](http://www.odva.org). NOTE: Because the technologies described in the CIP Networks Library may be applied in many diverse situations and in conjunction with products and systems from multiple vendors, the user and those responsible for specifying these technologies must determine for themselves their suitability for the intended use. ALL INFORMATION PROVIDED BY ODVA IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, AND ODVA AND ITS MEMBERS, PARTICIPANTS, SPECIAL INTERESTS GROUPS, EXECUTIVE DIRECTOR AND BOARD OF DIRECTORS EXPRESSLY DISCLAIM ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR OR INTENDED PURPOSE, OR ANY OTHER WARRANTY OTHERWISE ARISING OUT OF THE SPECIFICATIONS. ODVA AND ITS MEMBERS, PARTICIPANTS, SPECIAL INTERESTS GROUPS, EXECUTIVE DIRECTOR AND BOARD OF DIRECTORS DO NOT WARRANT THAT USE OF THE SPECIFICATIONS (INCLUDING, WITHOUT LIMITATION, THE MANUFACTURE, DISTRIBUTION AND SALE OF PRODUCTS THAT COMPLY WITH THE SPECIFICATIONS) WILL BE ROYALTY-FREE. The user should always verify interconnection requirements to and from other equipment, and confirm installation and maintenance requirements for their specific application. IN NO EVENT SHALL ODVA, ITS OFFICERS, DIRECTORS, MEMBERS, AGENTS, LICENSORS, OR AFFILIATES BE LIABLE TO YOU, ANY CUSTOMER, OR THIRD PARTY FOR ANY DAMAGES, DIRECT OR INDIRECT, INCLUDING BUT NOT LIMITED TO LOST PROFITS, DEVELOPMENT EXPENSES, OR ANY OTHER DIRECT, INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES.

The following are trademarks of ODVA:

CIP  
CIP Motion  
CIP Safety  
CIP Safety CONFORMANCE TESTED  
CIP Sync  
DeviceNet  
DeviceNet CONFORMANCE TESTED  
CompoNet  
CompoNet CONFORMANCE TESTED  
ControlNet  
ControlNet CONFORMANCE TESTED  
EtherNet/IP  
EtherNet/IP CONFORMANCE TESTED

All other trademarks referenced herein are property of their respective owners.

#### SITE SUBSCRIPTION

- The Final Specification, of which this volume and edition of The CIP Networks Library is a part, is provided on an annual subscription basis to this Licensed Vendor Member, as defined by its unique Vendor ID for the technology contained in the Final Specification (“YOU”), pursuant to your Terms of Usage Agreement with ODVA, Inc. (ODVA) for the technology contained in the Final Specification.
- This subscription is a site subscription, and this subscription, along with your membership in ODVA, must be renewed annually in order to maintain continuing rights to the site subscription as allowed under your Terms of Usage Agreement.
- This site subscription permits access to the electronic files for this volume contained on the distribution CD by multiple users who are your employees and on-site contracted individuals performing typical employee functions on a contract basis (“Authorized Users”). YOU shall ensure that, if access to these files is given to contractors, the contractors can fulfill the obligations of your Terms of Usage Agreement for the ODVA technology contained in the Final Specification. YOU shall not knowingly permit anyone other than Authorized Users to access these files.
- This site subscription permits access to the electronic files contained on the distribution CD for this volume via the original CD on which this volume is distributed or via an electronic copy of this volume placed on your single, secure intranet site. If and when the electronic files are posted on your intranet site, YOU shall relocate the original CD to secure storage for use only as a system back-up for the electronic files posted on your intranet site.
- The possession of or subscription to this Final Specification does not, by itself, convey any right to use or reproduce any portion of the Final Specification or to make, have made, use, import, offer to sell, sell, lease, market, or otherwise distribute or dispose of any products contemplated by the Final Specification, and you are hereby notified that the products contemplated by this Final Specification might be covered by valid patents or copyrights of ODVA, its members or other licensors. The necessary licenses to use or reproduce portions of the Final Specification for use in products, or to make, have made, use, import, offer to sell, sell, lease, market, or otherwise distribute and dispose of such products may be obtained only from ODVA through its Terms of Usage Agreement, available through the ODVA web site. This license requirement applies equally (a) to devices that completely implement the Final Specification with a network port that can be issued a Declaration of Conformity (“CIP Network Devices”), (b) to components of such CIP Network Devices to the extent they implement portions of the Final Specification, and (c) to enabling technology products, such as any CIP Network protocol stack, designed for use in CIP Network Devices to the extent they implement portions of the Final Specification. Contact ODVA for a Terms of Usage Agreement if you are not already licensed.
- Any other distribution and all other electronic copies, intranet or internet postings, and any printed copies are prohibited. Printed copies of this volume may be purchased via the order form available through the ODVA web site at [www.odva.org](http://www.odva.org).
- Notwithstanding anything to the contrary herein, if in connection with bookmarking a page of the Final Specification it is reasonably necessary for an Authorized User to possess a copy of the Final Specification on the computer on which such page is bookmarked, then you may possess such copy, provided that (i) such copy is not accessible to anyone other than the Authorized User, (ii) such copy is not retained for any longer than reasonably necessary to use the bookmarked page and in any event no longer than the term of this subscription, (iii) use of such copy is limited to the uses permitted in this site subscription and (iv) such copy is not transmitted or otherwise distributed to any other person, computer or device.

This page is intentionally left blank



## The CIP Networks Library: Volume 3

### DeviceNet Adaptation of CIP

#### Table of Contents

<b>Revisions</b>	- Summary of Changes in this Edition
<b>Preface</b>	- Organization of CIP Networks Specifications - The Specification Enhancement Process
<b>Chapter 1</b>	- Introduction to DeviceNet
<b>Chapter 2</b>	- DeviceNet Messaging Protocol
<b>Chapter 3</b>	- DeviceNet Communications
<b>Chapter 4</b>	- CIP Object Model
<b>Chapter 5</b>	- Object Library
<b>Chapter 6</b>	- Device Profiles
<b>Chapter 7</b>	- Electronic Data Sheets
<b>Chapter 8</b>	- Physical Layer
<b>Chapter 9</b>	- Indicators and Middle Layers
<b>Chapter 10</b>	- Bridging and Routing
<b>Appendix A</b>	- Explicit Messaging Services
<b>Appendix B</b>	- Status Codes
<b>Appendix C</b>	- Data Management
<b>Appendix D</b>	- Engineering Units

## Revisions

The CIP Networks Library Volume 3: DeviceNet Adaptation of CIP Edition 1.10 contains the following changes from the previous Edition. Please see the change bars on the pages noted here for specific modifications. Note: Some of the pages within the ranges noted may not contain any changes.

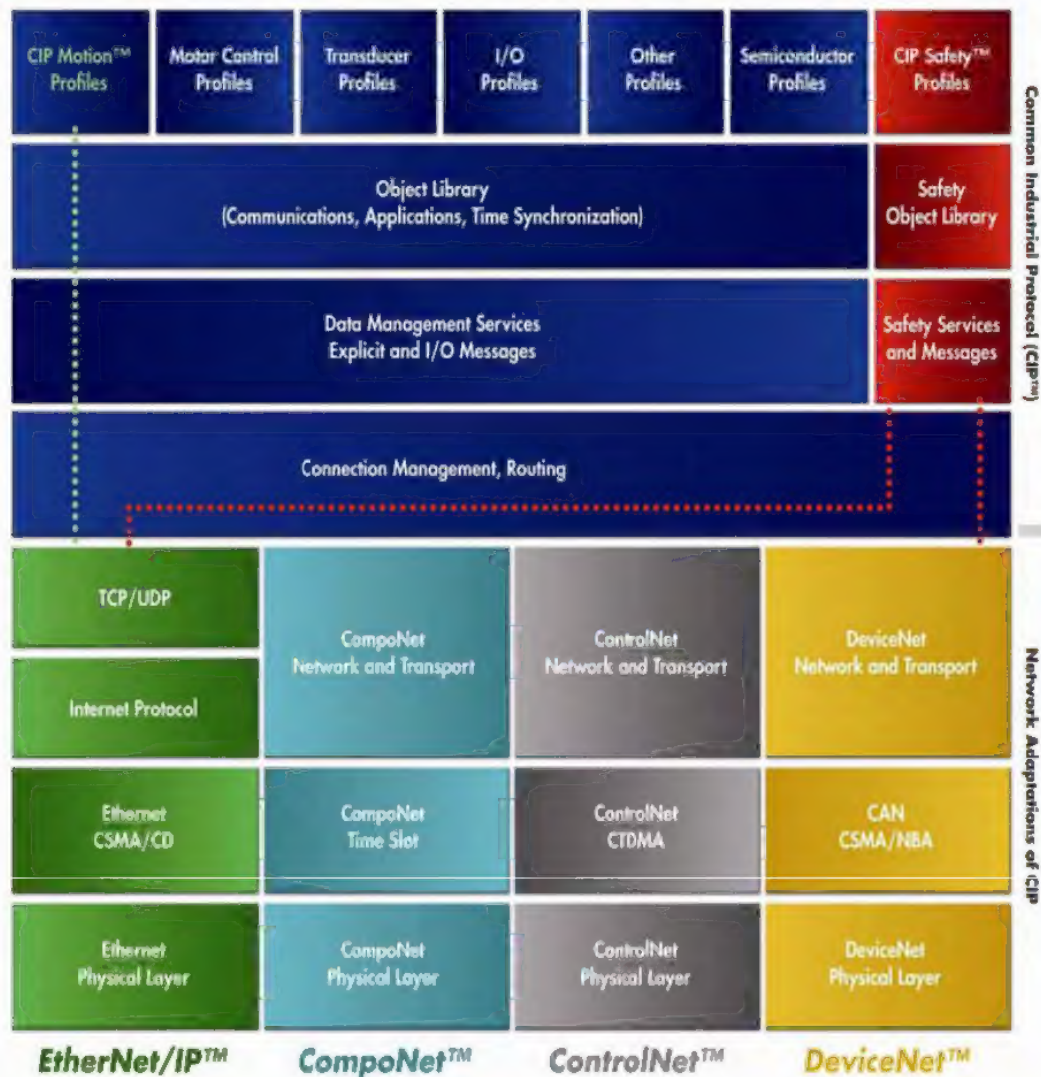
Chapt-Sect	Pages	Description
		<b>Missing NV Columns</b>
5-4.1	5-5	<ul style="list-style-type: none"><li>• Add NV column to Class attributes table</li></ul>
5-4.3	5-6	<ul style="list-style-type: none"><li>• Add NV column to Instance attributes table</li></ul>
		<b>Keyable Generic Device Type</b>
7-4	7-5	<ul style="list-style-type: none"><li>• Changed ProdType from 0 to 43 in Figure 7-4.1. Example of Partial EDS</li></ul>
		<b>MS LED Ambiguity</b>
9-2.2	9-4	<ul style="list-style-type: none"><li>• Changes to Minor Fault state in Table 9-2.1</li></ul>

## Preface

### Organization of the CIP Networks Specifications

Today, four networks - DeviceNet™, ControlNet™, EtherNet/IP™ and CompoNet™ - use the Common Industrial Protocol (CIP) for the upper layers of their network protocol. For this reason, ODVA manages and distributes the specifications for CIP Networks in a common structure to help ensure consistency and accuracy in the management of these specifications.

The following diagram illustrates the organization of the library of CIP Network specifications. In addition to CIP Networks, CIP Safety™ consists of the extensions to CIP for functional safety.



This common structure presents CIP in one volume with a separate volume for each network adaptation of CIP. The specifications for the CIP Networks are two-volume sets, paired as shown below.

The EtherNet/IP specification consists of:

Volume 1: Common Industrial Protocol (CIP™)

Volume 2: EtherNet/IP Adaptation of CIP

The DeviceNet specification consists of:

Volume 1: Common Industrial Protocol (CIP™)

Volume 3: DeviceNet Adaptation of CIP

The ControlNet specification consists of:

Volume 1: Common Industrial Protocol (CIP™)

Volume 4: ControlNet Adaptation of CIP

The CompoNet specification consists of:

Volume 1: Common Industrial Protocol (CIP™)

Volume 6: CompoNet Adaptation of CIP

The specifications for CIP Safety™ is distributed in a single volume:

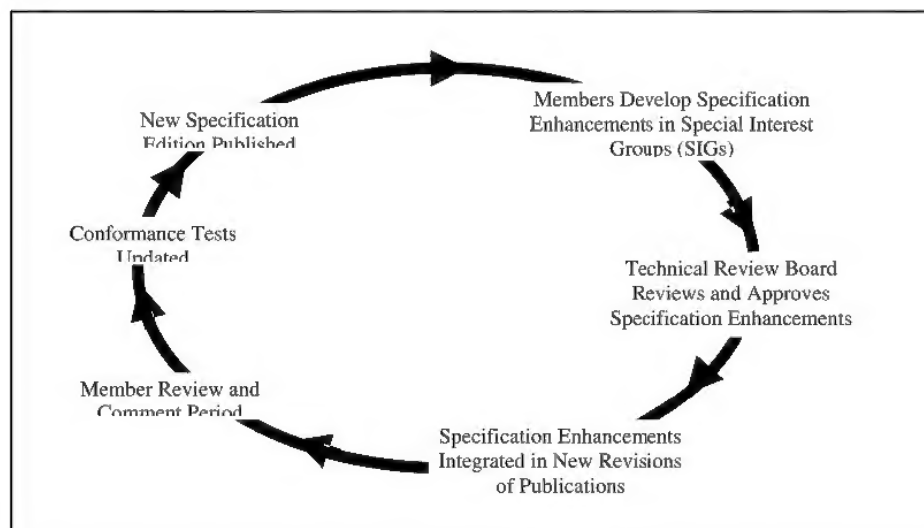
Volume 5: CIP Safety

The specification for integrating Modbus Devices is distributed in a single volume:

Volume 7: Integration of Modbus Devices into the CIP Architecture

#### Specification Enhancement Process

The specifications for CIP Networks are continually being enhanced to meet the increasing needs of users for features and functionality. ODVA has implemented a Specification Enhancement Process in order to ensure open and stable specifications for all CIP Networks. This process is ongoing throughout the year for each CIP Network Specification as shown in the figure below. New editions of each CIP Network specification are published on a periodic basis.



## **Volume 3: DeviceNet Adaptation of CIP**

# **Chapter 1: Introduction to DeviceNet**



## **Contents**

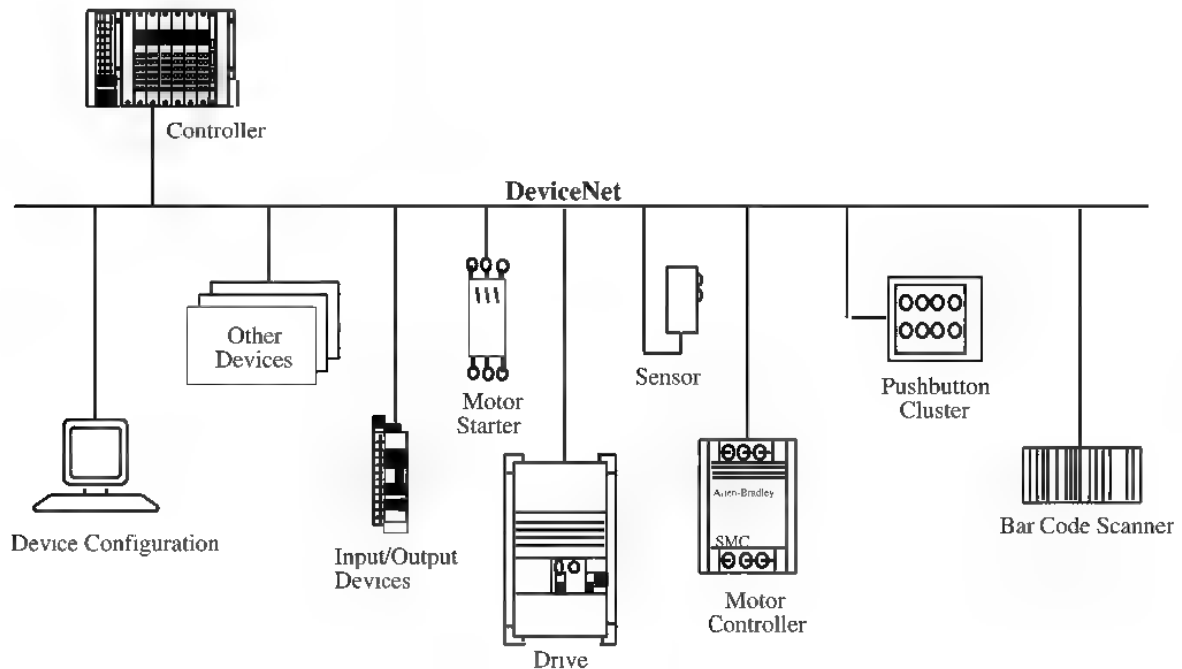
1-1	Introduction.....	3
1-2	DeviceNet Characteristics.....	4
1-3	Object Modeling .....	6
1-4	DeviceNet MAC ID Range .....	7
1-5	DeviceNet Object Model .....	7
1-6	Normative References.....	9

## 1-1 Introduction

DeviceNet™ is a low-level network that provides connections between simple industrial devices (sensors, actuators) and higher-level devices (controllers). DeviceNet is based on the Common Industrial Protocol (CIP), and shares all the common aspects of CIP with adaptations to fit the message frame size of DeviceNet.

Figure 1-1.1 provides an example of a typical DeviceNet network.

**Figure 1-1.1 Example DeviceNet Communication Link**



DeviceNet provides:

- A cost effective solution to low-level device networking
- Access to intelligence present in low-level devices
- Master/Slave and Peer-to-Peer capabilities

DeviceNet has two primary purposes:

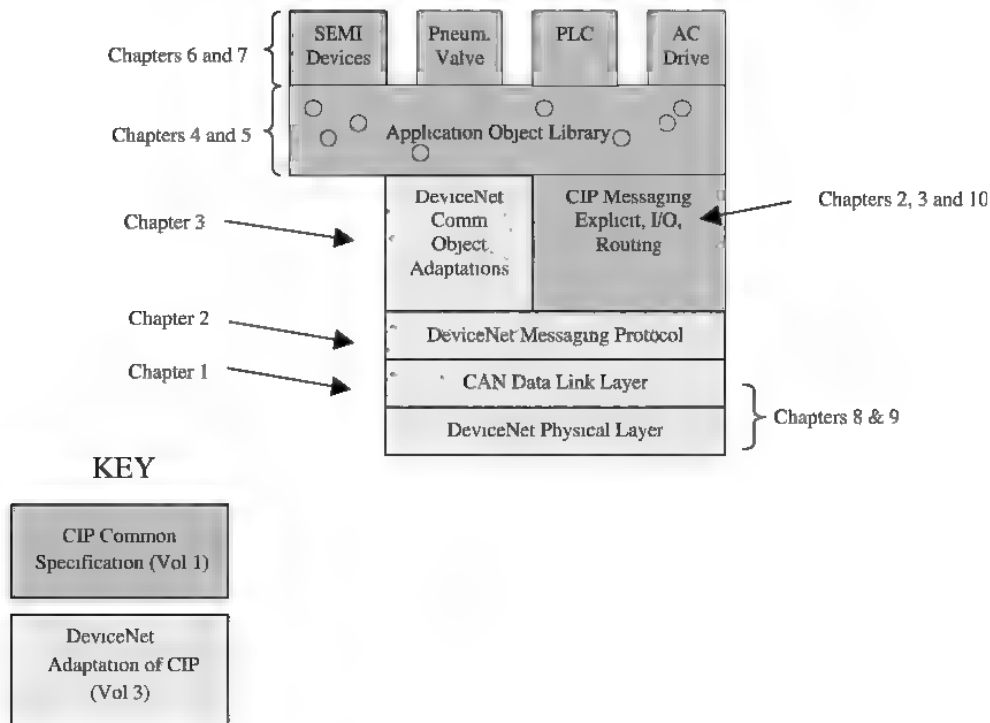
- Transport of control-oriented information associated with low-level devices

Transport of other information, which is indirectly related to the system being controlled, such as configuration parameters.

This chapter is the Introduction to DeviceNet. The following drawing shows the relationship of these chapters to each other and to the CIP Common Specification (published separately by ODVA and ControlNet International). Both this specification (Volume 3) and the CIP Common specification (Volume 1) are required to completely specify a DeviceNet product.

Figure 1-1.2 shows the relationship between the various parts of the DeviceNet specification. As shown in the figure, the darker sections are predominately documented by the CIP Common specification (Volume 1). The corresponding chapters of the DeviceNet Adaptation of CIP (Volume 3) supplements or modifies these chapters of the CIP Common specification in some areas. The lightly shaded sections are predominately documented by volume 3. These chapters contain information applicable specifically to DeviceNet devices, but no necessarily to those of other CIP networks, (for example, EtherNet/IP or ControlNet).

**Figure 1-1.2 Document Organization Overview**



## 1-2 DeviceNet Characteristics

The list below presents a summary of the Physical/Media specific characteristics of DeviceNet:

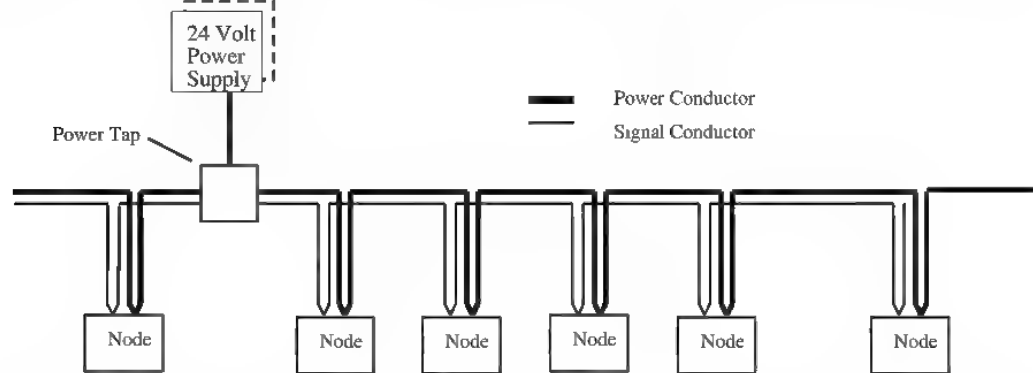
- Trunkline-dropline configuration
- Support for up to 64 nodes
- Node removal without severing the network
- Simultaneous support for both network-powered (sensors) and self-powered (actuators) devices
- Use of sealed or open-style connectors
- Protection from wiring errors
- Selectable data rates of 125k baud, 250k baud, and 500k baud

**Table 1-2.1 Data Rate versus Drop Length**

Data Rate	Trunk Distance	Drop Length	
		Maximum	Cumulative
125k baud	500 meters (1640 ft.)	6 meters (20 ft.)	156 meters (512 ft.)
250k baud	250 meters (820 ft.)		78 meters (256 ft.)
500k baud	100 meters (328 ft.)		39 meters (128 ft.)

- Adjustable power configuration to meet individual application needs
- High current capability (up to 16 amps per supply)
- Operation with off-the-shelf power supplies
- Power taps that allow the connection of several power supplies from multiple vendor that comply with DeviceNet standards
- Built-in overload protection
- Power available along the bus: both signal and power lines contained in the trunkline

**Figure 1-2.1 Power along the Bus**



The list below summarizes additional communication features provided by DeviceNet:

- Use of Controller Area Network (CAN) technology for Media Access Control and Physical Signaling
- Connection-based model to facilitate application to application communications
- Provisions for the typical request/response oriented network communications
- Provisions for the efficient movement of I/O data
- Fragmentation for moving larger bodies of information
- Duplicate MAC ID detection

## **1-3 Object Modeling**

DeviceNet makes use of the same abstract *object modeling* defined by CIP to describe:

- The suite of communication services available
- The externally visible behavior of a DeviceNet node
- A common means by which information within DeviceNet products is accessed and exchanged

Please see CIP Common, Chapter 1 for basic information on how DeviceNet uses Object Modeling. CIP Common, Chapter 1 defines the range of Class Ids, Attribute Ids and Service codes.



## **1-4 DeviceNet MAC ID Range**

DeviceNet MAC Ids are integer values in the range shown in Table 1-4.1.

**Table 1-4.1 MAC ID Ranges**

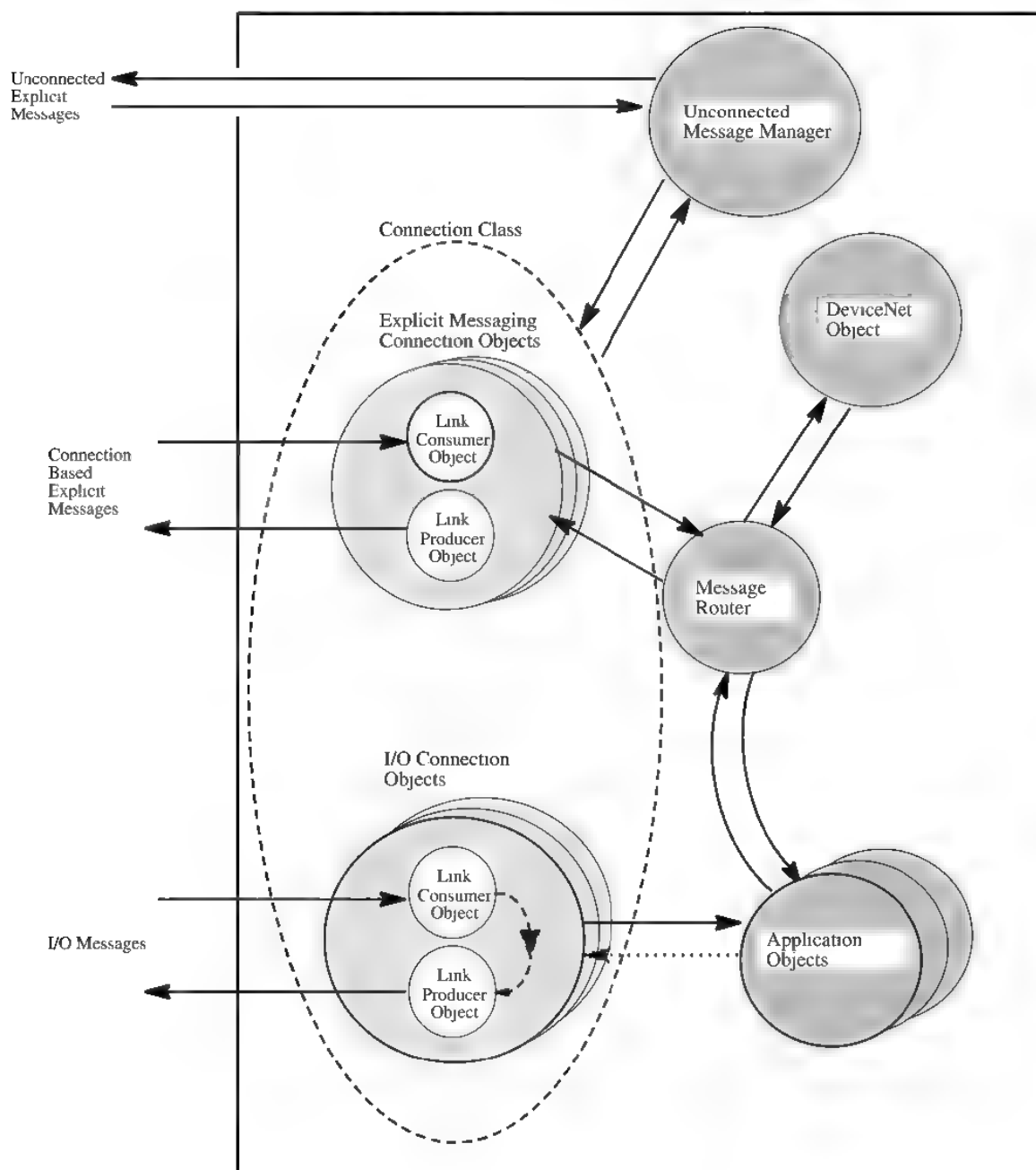
Range	Meaning
00 - 63 decimal	The MAC ID. The value 63 (decimal) is to be utilized upon initialization of a device (e.g. powerup) if another value has not been assigned.

## **1-5 DeviceNet Object Model**

Figure 1-5.1 illustrates the abstract object model of a DeviceNet product. Included are the following components:

- Unconnected Message Manager (UCMM - Processes DeviceNet Unconnected Explicit messages.
- Connection Class - Allocates and manages internal resources associated with both I/O and Explicit Messaging connections.
- Connection Object - Manages the communication-specific aspects associated with a particular application-to-application network relationship.
- DeviceNet Object - Provides the configuration and status of a physical DeviceNet network connection.
- Link Producer Object - Used by a Connection Object to transmit data onto DeviceNet.
- Link Consumer Object - Used by a Connection Object to receive data from DeviceNet.
- Message Router - Distributes Explicit Request Messages to the appropriate handler object.
- Application Objects - Implement the intended purpose of the product.

Figure 1-5.1 DeviceNet Device Object Model



## **1-6 Normative References**

The following standards contain provisions, which through references in this text constitute provisions of DeviceNet. At the time of publication, the editions indicated were valid.

- BOSCH CAN Specification - Version 2.0, Part A. 1991, Robert Bosch GmbH.
- ISO 11898: 1993 - Road vehicles - Interchange of digital information - Controller area network (CAN) for high-speed communication.

This page is intentionally left blank

## **Volume 3: DeviceNet Adaptation of CIP**

# **Chapter 2: DeviceNet Messaging Protocol**



## Contents

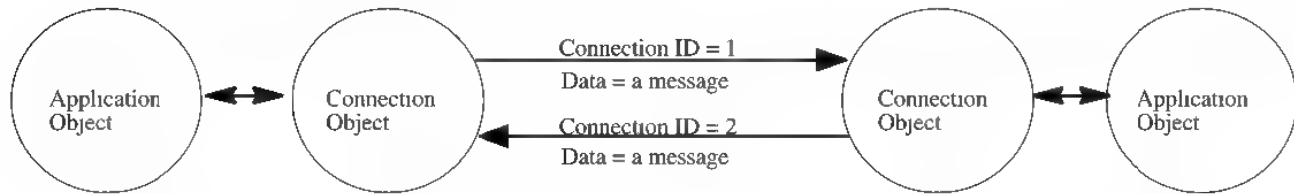
2-1	Introduction.....	4
2-2	DeviceNet's Use of the CAN Identifier Field .....	4
2-2.1	Message Group 1 .....	5
2-2.2	Message Group 2 .....	6
2-2.3	Message Group 3 .....	7
2-2.4	Message Group 4 .....	8
2-3	Network Access State Machine .....	9
2-3.1	State Transition Diagram & Event Matrix .....	9
2-3.2	Duplicate MAC ID Detection .....	11
2-3.3	Duplicate MAC ID Detection Protocol .....	12
2-3.3.1	Duplicate MAC ID Check Message Service Data .....	13
2-3.3.2	Duplicate MAC ID Check Example.....	14
2-3.4	Quick Connect.....	14
2-4	Connection Establishment Overview .....	15
2-4.1	Explicit Messaging Connections and the UCMM.....	15
2-4.2	I/O Connections .....	17
2-5	Predefined Master/Slave Connection Set.....	20
2-6	Client and Server Connection End-points.....	20
2-7	DeviceNet Messaging Protocol.....	21
2-7.1	Explicit Messaging.....	21
2-7.1.1	Message Header .....	22
2-7.1.2	Message Body .....	23
2-7.2	UCMM Services .....	24
2-7.2.1	Open Explicit Messaging Connection Request .....	25
2-7.2.1.1	Open Explicit Message Connection Request Data Frame Contents.....	25
2-7.2.1.2	Service Procedure .....	27
2-7.2.2	Open Explicit Messaging Connection Success Response ....	28
2-7.2.2.1	Open Explicit Message Connection Response Data Frame Contents .....	28
2-7.2.3	Open Explicit Messaging Connection Protocol Examples.....	30
2-7.2.4	Close Connection Request .....	35
2-7.2.4.1	Close Connection Request Data Frame Contents .....	35
2-7.2.5	Close Response .....	36
2-7.2.5.1	Close Connection Response Data Frame Contents ....	36
2-7.2.6	Close Connection Protocol Examples .....	36
2-7.2.7	Error Response .....	37
2-7.3	Connection Based Explicit Messaging.....	38
2-7.3.1	Explicit Request Data Frame (Message Body Format Values 0 – 3) .....	38
2-7.3.2	Explicit Request Data Frame Contents (Message Body Format value 4) .....	39
2-7.3.3	Success Response Explicit Message .....	39
2-7.3.4	Error Response Explicit Message .....	40
2-8	Input/Output Messaging.....	41
2-9	Fragmentation/Reassembly .....	41
2-9.1	Fragmentation Protocol.....	41
2-9.2	Unacknowledged Fragmentation.....	44
2-9.3	Acknowledged Fragmentation .....	49
2-9.3.1	Acknowledgment/Response Data Frame Contents .....	50
2-10	Explicit Messaging Client Application considerations .....	59
2-10.1	Request/Response Timeouts ..	59
2-10.2	Explicit Request Message Fragmentation Timeouts .....	60
2-11	Offline Connection Set.....	62
2-11.1	Offline Ownership.....	63
2-11.2	Offline Ownership Messages.....	64
2-11.2.1	Offline Ownership Request Message (Client Only).....	64

2-11.2.2	Offline Ownership Response Message (Client Only) .....	65
2-11.3	Communication Faulted Messages.....	66
2-11.3.1	Communication Faulted Message Protocols .....	66
2-11.4	Identify Communication Faulted Messages .....	69
2-11.4.1	Identify Request Message - Multicast protocol.....	69
2-11.4.2	Identify Response Message - Multicast protocol .....	69
2-11.4.3	Identify Request Message - Point-to-Point protocol .....	70
2-11.4.4	Identify Response Message - Point-to-Point protocol.....	71
2-11.5	Who Communication Faulted Request Message.....	71
2-11.6	Who Communication Response .....	72
2-11.7	Change MAC ID Communication Faulted Request Message.....	72
2-12	Device Heartbeat.....	73
2-12.1	Device Heartbeat Message .....	73
2-12.1.1	Data Frame Contents, Device Heartbeat Message .....	74
2-13	Device Shutdown Message .....	75
2-13.1	Device Shutdown Message .....	75
2-13.1.1	Data Frame Contents, Device Shutdown Message.....	75

## 2-1 Introduction

DeviceNet is a *connection*-based network. A DeviceNet *connection* provides a path between multiple applications. When a connection is established, the transmissions associated with that connection are assigned a **Connection ID (CID)**. If the connection involves a bi-directional exchange, then two Connection ID values are assigned. See Figure 2-1.1 Connections and Connection Ids.

Figure 2-1.1 Connections and Connection Ids



This chapter explains DeviceNet's use of the CAN Identifier Field, and defines the steps involved in the dynamic establishment of I/O and Explicit Messaging Connections.

## 2-2 DeviceNet's Use of the CAN Identifier Field

The 11 CAN Identifier bits available on DeviceNet are subdivided into four separate message groups: Group 1, Group 2, Group 3, and Group 4.

Figure 2-2.1 DeviceNet's Use of the CAN Identifier Field

IDENTIFIER BITS											HEX RANGE	IDENTITY USAGE
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MAC ID						000 – 3ff	Message Group 1
1	0	MAC ID						Group 2 Message ID			400 - 5ff	Message Group 2
1	1	Group 3 Message ID			Source MAC ID						600 – 7bf	Message Group 3
1	1	1	1	1	Group 4 Message ID (0 2f)						7c0 – 7ef	Message Group 4
1	1	1	1	1	1	1	X	X	X	X	7f0 – 7ff	Invalid CAN Identifiers
10	9	8	7	6	5	4	3	2	1	0		

With respect to Connection Based Messages, the Connection ID is placed within the CAN Identifier Field. With this in mind, Figure 2-2.1 also describes the components of a DeviceNet Connection ID.

As Figure 2-2.1 indicates, the CAN Identifier Field on DeviceNet contains the following components:

- Message ID** - Identifies a message within a Message Group inside a particular end-point. The Message ID facilitates the establishment of multiple Connections within a single Message Group inside a specific end-point. When a Connection is established, the end-points utilize a Message ID in combination with a MAC ID to generate a Connection ID. The resulting Connection ID is specified in the CAN Identifier Field associated with related transmissions. This is described in detail throughout the remainder of the specification. Note that Group 2 and Group 3 predefine uses for certain Message IDs.

- **Source MAC ID** - The MAC ID assigned to the transmitting node. Groups 1 and 3 require the specification of a Source MAC ID within the CAN Identifier Field.
- **Destination MAC ID** - The MAC ID assigned to the receiving device. Message Group 2 allows the specification of either Source or Destination within the MAC ID portion of the CAN Identifier Field

The Message Groups were designed using the following concepts:

- Present a solution in which bus access priority is not based solely upon a node's MAC ID, and is as *distributed* as possible. This is realized within Groups 1 and 3.
- Provide a solution that takes into consideration the limited filtering capabilities of many CAN chips. This is realized within Group 2.

Both Explicit Messaging and I/O Connections can be established in Message Groups 1, 2, 3 and Offline Connection Set messages in Group 4. Because of the arbitration scheme defined by CAN, Group 1 messages are higher priority than Group 2 and Group 2 messages are higher priority than Group 3 messages. This prioritization must be taken into consideration when establishing connections. Establishing a Group 1 Explicit Messaging Connection may cause delays in I/O exchanges that are also occurring in Group 1.

The following sections describe the four Message Groups in detail. The shaded portions of the Identifier Bits illustrate the bits that distinguish the Message Groups.

## 2-2.1 Message Group 1

Figure 2-2.2 provides an expanded view of the Group 1 Identifiers and illustrates that DeviceNet does not predefine a use for any of the Group 1 Message IDs. The Group 1 Message ID delineates the various Group 1 messages exchanged by a particular end-point.

**Figure 2-2.2 Message ID Use within Group 1**

IDENTIFIER BITS											MESSAGE ID MEANING
10	9	8	7	6	5	4	3	2	1	0	
0	Group 1 Message ID				Source MAC ID						Group 1 Messages
0	0	0	0	0	Source MAC ID						Group 1 Message Identifier
0	0	0	0	1	Source MAC ID						
0	0	0	1	0	Source MAC ID						
0	0	0	1	1	Source MAC ID						
0	0	1	0	0	Source MAC ID						
0	0	1	0	1	Source MAC ID						
0	0	1	1	0	Source MAC ID						
0	0	1	1	1	Source MAC ID						
0	1	0	0	0	Source MAC ID						
0	1	0	0	1	Source MAC ID						
0	1	0	1	0	Source MAC ID						
0	1	0	1	1	Source MAC ID						
0	1	1	0	0	Source MAC ID						
0	1	1	0	1	Source MAC ID						
0	1	1	1	0	Source MAC ID						
0	1	1	1	1	Source MAC ID						

Within Group 1 transmissions, bus access priority is distributed evenly among all the devices on the network. When two or more Group 1 Messages are arbitrating for CAN bus access, the message with a numerically lower Group 1 Message ID value will win the arbitration and gain bus access.

For example, device #20, message\_id = 2 wins arbitration against device #5, message\_id = 6.

If two or more Group 1 Messages whose Group 1 Message ID values are equal arbitrate for the bus, then the transmission from the device with the lower MAC ID value will win arbitration.

For example, device #2, message\_id = 5 wins arbitration against device #3, message\_id = 5. Thus, an evenly distributed priority scheme with 16 different levels is presented within Group 1.

## 2-2.2 Message Group 2

Figure 2-2.3 provides an expanded view of the Group 2 Identifiers. The Group 2 Message ID delineates the various Group 2 messages exchanged by a particular end-point. The exception to this use is Group 2 Message ID values 6 and 7:

- DeviceNet predefines a set of Connections to facilitate the communications observed in a Master/Slave Application (see Chapter 3-6). This definition reserves Group 2 Message ID value 6.
- Group 2 Message ID 7 is reserved for use in the detection of nodes that have been assigned identical MAC IDs (see Section 2-3, Network Access State Machine).

**Important:** Use of Group 2 Message ID values 6 and 7 is reserved by DeviceNet.

Figure 2-2.3 Message ID Use Within Group 2

IDENTIFIER BITS										MESSAGE ID MEANING	
10	9	8	7	6	5	4	3	2	1		
1	0	MAC ID					Group 2 Message ID		Group 2 Messages		
1	0	MAC ID					0	0			
1	0	MAC ID					0	0	1	Group 2 Message Identifier	
1	0	MAC ID					0	1	0		
1	0	MAC ID					0	1	1		
1	0	MAC ID					1	0	0		
1	0	MAC ID					1	0	1		
1	0	Destination MAC ID					1	1	0	Reserved for Predefined Master/Slave Connection Management	
1	0	Destination MAC ID					1	1	1	Duplicate MAC ID Check Message	

Within Group 2, the MAC ID can be either the transmitting node's MAC ID (**Source MAC ID**) or the receiving node's MAC ID (**Destination MAC ID**). When establishing a connection across Group 2, the endpoints determine whether Source or Destination MAC ID should be specified.

Within Group 2 transmissions, bus access priority depends upon the value MAC ID placed within the MAC ID portion of the Identifier. When 2 or more Group 2 transmissions arbitrate for the CAN bus, the message whose MAC ID component is numerically lower will gain bus access.

For example, Group 2 MAC ID = 0 wins arbitration against Group 2 MAC ID = 1. If two or more devices are attempting to transmit a Group 2 Message specifying the same value within the MAC ID portion of the Group 2 Identifier, then the transmission specifying the numerically lowest Group 2 Message ID value gains bus access.

### 2-2.3 Message Group 3

Figure 2-2.4 below provides an expanded view of the Group 3 Identifiers. The Group 3 Message ID delineates the various Group 3 messages exchanged by a particular end-point. The exception to this use is Group 3 Message ID values 5, 6 and 7:

- Group 3 Message ID 5 is used when sending responses associated with Unconnected Explicit Messaging Requests as well as Device Heartbeat and Device Shutdown Messages.
- Group 3 Message ID 6 is used when sending Unconnected Explicit Messaging Requests.
- Group 3 Message ID value 7 is invalid and is not used.

Messages that dynamically establish Explicit Messaging Connections are transmitted under Group 3 and place the value 5 (responses) and/or 6 (requests) in the Group 3 Message ID portion of the CAN Identifier Field. These messages are referred to as **Unconnected** Explicit Messages. Unconnected Explicit Messages are handled/processed by the Unconnected Message Manager (UCMM).

**Important:** Use of Group 3 Message ID values 5, 6 and 7 is reserved by DeviceNet.

**Figure 2-2.4 Message ID Use Within Group 3**

IDENTIFIER BITS											MESSAGE ID MEANING
10	9	8	7	6	5	4	3	2	1	0	
1	1	Group 3 Message ID			Source MAC ID						Group 3 Messages
1	1	0	0	0	Source MAC ID						Group 3 Message Identifier
1	1	0	0	1	Source MAC ID						
1	1	0	1	0	Source MAC ID						
1	1	0	1	1	Source MAC ID						
1	1	1	0	0	Source MAC ID						
1	1	1	0	1	Source MAC ID						Unconnected Explicit Response Messages
1	1	1	1	0	Source MAC ID						Unconnected Explicit Request Messages

Within Group 3 transmissions, bus access priority is distributed evenly among all the nodes on the network. When two or more Group 3 Messages are arbitrating for CAN bus access, the message whose Group 3 Message ID value is numerically lower will win the arbitration and gain bus access.

For example, device #20, message\_id = 2 wins arbitration against device #5, message\_id = 4.

If two or more Group 3 Messages whose Group 3 Message ID values are equal arbitrate for the bus, then the transmission from the device with the lower MAC ID value will win arbitration.

For example, device #2, message\_id = 4 wins arbitration against device #3, message\_id = 4. Thus, an evenly distributed priority scheme with five (5) different levels is presented within Group 3.

## 2-2.4 Message Group 4

The figure below provides an expanded view of the Group 4 Identifiers:

Figure 2-2.5 Message ID Usage within Group 4

IDENTIFIER BITS											IDENTITY
10	9	8	7	6	5	4	3	2	1	0	USAGE
1	1	1	1	1	Group 4 Message ID						Group 4 Messages
1	1	1	1	1	0 - 2B						Reserved Group 4 Messages
1	1	1	1	1	2C						Communication Faulted Response Message
1	1	1	1	1	2D						Communication Faulted Request Message
1	1	1	1	1	2E						Offline Ownership Response Message
1	1	1	1	1	2F						Offline Ownership Request Message
1	1	1	1	1	1	1	X	X	X	X	Invalid CAN Identifiers

The identifiers, which are not reserved, shall be used for system administration purposes. Currently message dialogs are defined which allow a tool to recover nodes, which have gone off-line due to having the same network address of other nodes on the subnet (see section 2-11, **Offline Connection Set**). It also allows a user to flash a node's LED to identify the specific node a tool is currently manipulating.

Group 4 Message ID 2C is used by *Communication Faulted* nodes when producing *Communication Faulted Response Messages*.

Group 4 Message ID 2D is used by client tools when producing *Communication Faulted Request Messages*.

Group 4 Message ID 2E is used by client tools when producing *Offline Ownership Response Messages*.

Group 4 Message ID 2F is used by client tools when producing *Offline Ownership Request Messages*.

Group 4 Message IDs 2C-2F are collectively referred to as the *Offline Connection Set Messages*.

For details on the usage of these identifiers see section 2-11, **Offline Connection Set**.

## 2-3 Network Access State Machine

This section defines the Network Access State Machine that every DeviceNet product must implement. The Network Access State Machine describes the following:

- Tasks that must be performed prior to communicating on DeviceNet
- Network events that affect a product's capability to communicate on DeviceNet.

### 2-3.1 State Transition Diagram & Event Matrix

Figure 2-3.1 provides a general overview of the Network Access State Machine.

Figure 2-3.1 Network Access State Transition Diagram

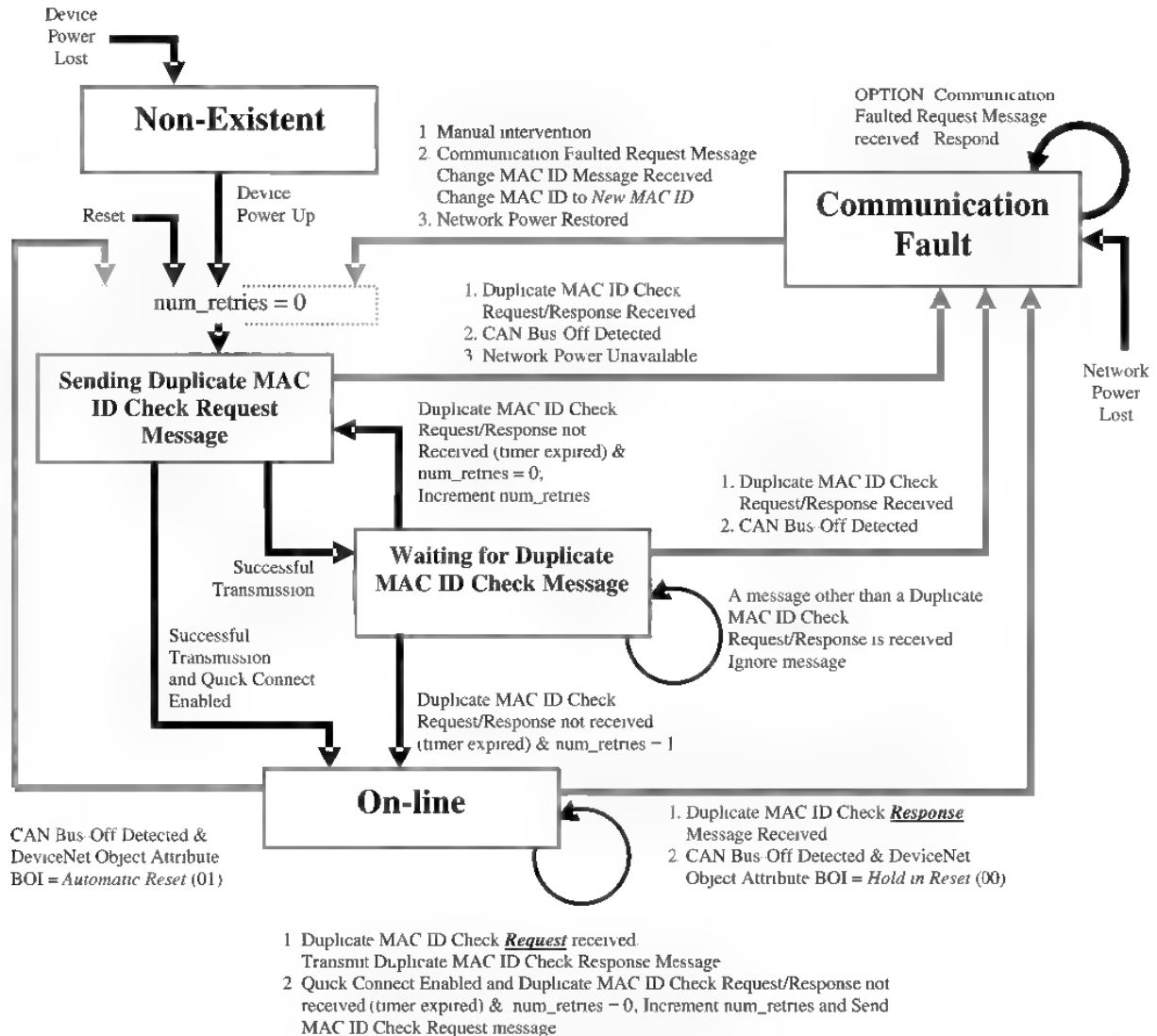


Table 2-3.1 provides a detailed State Event Matrix for the Network Access State Machine. Implementations should be based on the information in Table 2-3.1.



Table 2-3.1 Network Access State Event Matrix

Event	State			
	Sending Duplicate MAC ID Check Request	Waiting For Duplicate MAC ID Check Message	On-Line	Communication Fault
Successful transmission of the Duplicate MAC ID Check Request Message <sup>1</sup>	Activate 1 second timer. If Quick Connect is enabled transition to <b>On-Line</b> , otherwise transition to <b>Waiting For Duplicate MAC ID Check Message</b>	Not applicable	Not applicable	Not applicable
CAN Bus-Off detected	CAN chip held in reset. Transition to the <b>Communication Fault</b> state	CAN chip held in reset. Transition to the <b>Communication Fault</b> state	Access the DeviceNet Object's BOI attribute. If the BOI attribute indicates that the CAN Chip should be held in reset, then transition to <b>Communication Fault</b> . If the BOI attribute indicates that the CAN Chip should be automatically reset, then 1) reset the CAN Chip, 2) request the transmission of the Duplicate MAC ID Check <b>Request</b> Message, and 3) transition to the <b>Sending Duplicate MAC ID Check Request</b> state	Not applicable
Duplicate MAC ID Check Request Message Received	Duplicate MAC ID Detected. Transition to the <b>Communication Fault</b> state	Duplicate MAC ID Detected. Transition to the <b>Communication Fault</b> state	Transmit the Duplicate MAC ID Check <b>Response</b> Message	Discard the message
Duplicate MAC ID Check Response Message Received	Duplicate MAC ID detected. Transition to the <b>Communication Fault</b> state.	Duplicate MAC ID Detected. Transition to the <b>Communication Fault</b> state.	Duplicate MAC ID Detected. Transition to the <b>Communication Fault</b> state.	Discard the message
1 second Duplicate MAC ID Check Message timer expires	Not applicable	If this is the first timeout, then request the transmission of the Duplicate MAC ID Check <b>Request</b> Message again and transition to <b>Sending Duplicate MAC ID Check Request</b> . If this is the second consecutive timeout, then transition to <b>On-Line</b> .	If Quick Connect is enabled request the transmission of the Duplicate MAC ID Check <b>Request</b> Message again.	Not applicable
Internal message transmission request	Return internal error	Return internal error	Transmit the message	Return internal error
A message other than Duplicate MAC ID Check Request/Response or a Communication Faulted Request Message is received	Discard message	Discard message	Process the received message as appropriate	Discard the message

Event	State			
	Sending Duplicate MAC ID Check Request	Waiting For Duplicate MAC ID Check Message	On-Line	Communication Fault
A Communication Faulted Request message is received	Discard message	Discard message	Discard message	Process the received message as appropriate

- I The CAN Fault Confinement State Machine considers the possibility that during system start-up/wake-up, only one node may be present on the network. If this node transmits a message, it will experience an Acknowledgement Error and CAN silicon will automatically repeat the message. In this situation the node will transition to Error Passive **but not Bus-Off**. For this reason, the **ONLY** event that signifies an unsuccessful transmission of a Duplicate MAC ID Check Message (Request or Response) is the **Bus-Off** event. If an *error passive* or *error warning* indication is received during the transmission of a Duplicate MAC ID Check Message, then ignore it, as it has no effect on the Duplicate MAC ID Detection State Machine.

**Important:** Recognize that only the following two events delivered from the CAN chip affect the Network Access State Machine:

- *Transmission Successfully Performed.* This indication is delivered when a message is successfully transmitted onto the network. This is the **ONLY** event that causes the transition from **Sending Duplicate MAC ID Check Request** to **Waiting For Duplicate MAC ID Check Message**.
- *Bus-Off Indication.* This indication informs host software that the CAN chip has transitioned to the *Bus-Off* state. This causes the DeviceNet Object's BOI attribute to be accessed to determine the action to take.

An implementation must be prepared to manage these indications.

**Important:** A *message received* event illustrated in Figure 2-3.1 and Table 2-3.1 means that the message has passed through all appropriate screening logic (both CAN Chip provided and software) such that it will be processed. With respect to Duplicate MAC ID Check Messages, this means that the Destination MAC ID in the Group 2 Identifier Field is equal to the node's MAC ID.

## 2-3.2 Duplicate MAC ID Detection

The main step involved in the Network Access State Machine is the execution of a Duplicate MAC ID Detection algorithm. Each physical attachment to DeviceNet must be assigned a unique MAC ID. The configuration of this MAC ID will involve human intervention, and it is inevitable that two modules on the same link will be assigned the same MAC ID. Because the MAC ID is involved in defining the meaning of a DeviceNet transmission, **ALL** DeviceNet modules are **required** to participate in this duplicate MAC ID detection algorithm.

A DeviceNet module must receive and process any Duplicate MAC ID Check Message specifying its MAC ID in the Message Group 2 Identifier Field.

**Important:** The Duplicate MAC ID Check Request Message must be transmitted **two consecutive times** without receiving a subsequent Duplicate MAC ID Check Request **or** Response Message before transitioning to **On-Line**. The exception to this rule is when Quick Connect is enabled which allows a device to transition to the On-Line state after transmitting the first Duplicate MAC ID Check Request Message (see 2-3.4 Quick Connect).

**Important:** After transmitting a Duplicate MAC ID Check Request Message, a module waits at least **one (1) second** before timing out and taking the appropriate action defined by the Network Access State Machine.

The following pseudo code details reception logic relative to the duplicate MAC ID detection algorithm.

```

Message Received .....
If((Duplicate MAC ID Check Message) && (Destination MAC ID == My MAC ID))
    If (it is a response message)
        execute duplicate MAC ID detection logic
        transition to the communication fault state
    else If(my_state == On-Line) // duplicate mac id check request
        transmit Duplicate MAC ID Check Response Message
    else // request & I'm not on line
        execute duplicate MAC ID detection logic
        transition to the communication fault state
Else If (my_state == On Line)
    process received frame as appropriate
Else
    ignore received message // not on line yet
    
```

As defined in section 2-3.3, the Vendor Id and Serial Number integers are included in the Duplicate MAC ID Check Request/Response Messages. The presence of the Vendor ID/Serial Number information ensures that if two or more nodes with the same MAC ID attempt to execute the procedure at the same instant in time, network errors will result during transmission of the Duplicate MAC ID Check Messages. The network errors prohibit the devices from successfully transitioning to On-Line.

### 2-3.3 Duplicate MAC ID Detection Protocol

This section defines the protocol associated with the Duplicate MAC ID Check algorithm covered in the previous section.

A special Message ID value has been defined within Group 2 to specify the Duplicate MAC ID Check Message.

**Table 2-3.2 Duplicate Duplicate MAC ID Check CAN Identifier Field**

IDENTIFIER BITS											MESSAGE ID MEANING
10	9	8	7	6	5	4	3	2	1	0	
1	0	MAC ID						Group 2 Message ID			Group 2 Messages
1	0	Destination MAC ID						1	1	1	Duplicate MAC ID Check Message

The Data Field associated with the Duplicate MAC ID Check Message has the following format:

Figure 2-3.2 Duplicate MAC ID Check Message Data Field Format

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	R/R	Physical Port Number						
1	Vendor ID						Low byte	
2							High byte	
3	Serial Number						Low byte	
4								
5								
6							High byte	

### 2-3.3.1 Duplicate MAC ID Check Message Service Data

**R/R Bit** - Request/Response flag. The value in this field indicates whether this is a Duplicate MAC ID Check *Request* or a *Response* message. Values are defined in Table 2-3.3.

Table 2-3.3 Duplicate MAC ID Check Req/Rsp Field Values

Value	Meaning
0	Request. A module attempting to perform the Duplicate MAC ID Check process sets the Req/Rsp Field to zero (0) when it transmits the Duplicate MAC ID Check Message.
1	Response. A module that receives a Duplicate MAC ID Check Message for which a response is to be returned sets the Req/Rsp Field to one (1) within the response message.

**Physical Port Number** - An identification value internally assigned to each physical attachment to DeviceNet. Products that implement multiple physical attachments to DeviceNet (e.g. multiple connectors) must assign each separate attachment a unique value within the range of 0 - 127decimal. Products that implement a single attachment (e.g. a single connector) should place the value zero (0) within this field.

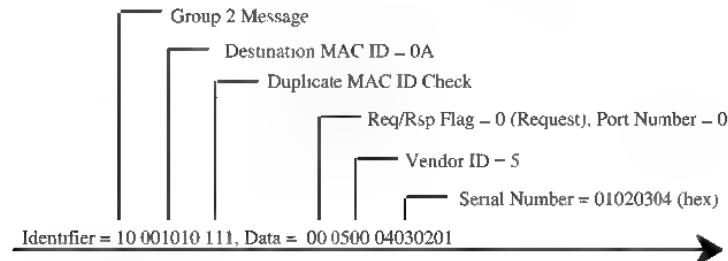
**Vendor ID** - A 16 bit integer field (UINT) containing the identification code assigned to the vendor of the device that is transmitting the message.

**Serial Number** - A 32 bit integer field (UDINT) containing the Serial Number assigned to the device by the Vendor.

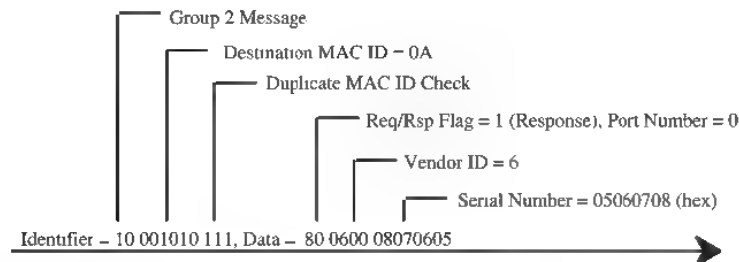
**Important:** All vendors that produce DeviceNet nodes will be assigned a Vendor Identification Code. In addition, each Vendor must assign every DeviceNet product a unique 32-bit serial number when manufacturing the product. All DeviceNet product manufacturing sites must be equipped to manage this process. The Serial Number need only be unique with respect to a particular vendor. Multiple vendors can, and most likely will, use the same Serial Number values.

### 2-3.3.2 Duplicate MAC ID Check Example

The following example illustrates the module with the assigned MAC ID of 0A, Vendor ID of 5, and Serial Number of 01020304 (hex) transmitting the Duplicate MAC ID Check Request Message.



Assume a device with the assigned MAC ID of 0A was already present on DeviceNet. The following example illustrates the message it would transmit because of the reception of the request message illustrated above.



### 2-3.4 Quick Connect

The Quick Connect feature is an option enabled on a node-by-node basis. When enabled, a device transitions to the OnLine state concurrently with sending the first Duplicate MACID Request message. The device is still required to execute the network STD, including going offline *anytime* a Duplicate MACID response message is received.

**Important:** Although this feature allows a device to begin participating in network activity faster, it is at the expense of a delay in the duplicate node detection algorithm. It is left up to the user to guarantee that no nodes exist with the same MAC ID *and* that no more than one Client device is configured to access the same device using the Predefined Master/Slave Connection Set. Bus errors may occur if either of these conditions exists.

This feature is enabled within a device through a non-volatile attribute in the DeviceNet object. A device shall have this feature disabled (attribute set to '0') as the factory default

## 2-4 Connection Establishment Overview

This section presents an overview of dynamically establishing both Explicit Messaging and I/O Connections.

### 2-4.1 Explicit Messaging Connections and the UCMM

Section 2-2.3, Message Group 3, discusses **Unconnected** Explicit Messaging. Unconnected Explicit Messages establish and manage Explicit Messaging Connections. Unconnected Request messages are specified by transmitting a Group 3 Message whose Message ID component is set to 6. The only valid Services that can be transmitted as Unconnected Explicit Request Messages are:

- Open Explicit Messaging Connection Request
- Close Connection Request

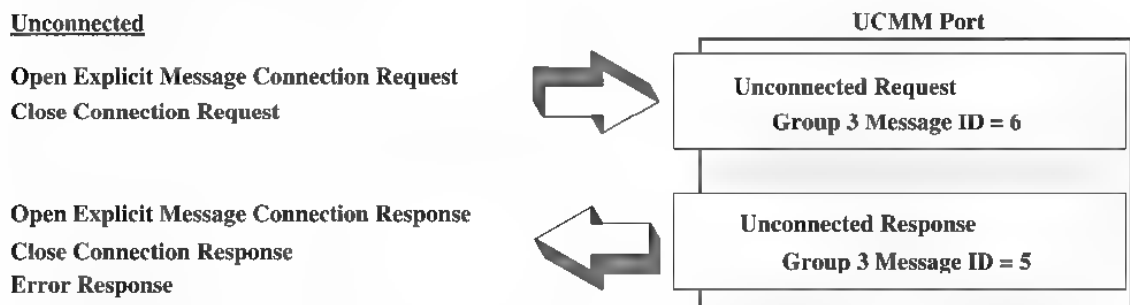
The messages listed above are **NEVER** transmitted as Connection Based Messages. See section 2-7.3, Connection Based Explicit Messaging.

Responses to Unconnected Explicit Requests are transmitted as Unconnected Response messages. Unconnected Response messages are specified by transmitting a Group 3 Message whose Message ID component is set to 5. The only valid Services that can be transmitted as Unconnected Explicit Response Messages are:

- Open Explicit Messaging Connection Response
- Close Connection Response
- Error Response
- Device Heartbeat Message
- Device Shutdown Message

The Unconnected Message Manager (UCMM) is responsible for processing Unconnected Explicit Requests and Responses. Support of the UCMM requires a device to screen for the Unconnected Explicit Request Message CAN Identifier from all possible Source MAC IDs. If a device transmits Unconnected Explicit Request Messages it must also screen for the Unconnected Explicit Response CAN Identifier from all possible Source MAC IDs.

Figure 2-4.1 UCMM Message Flow



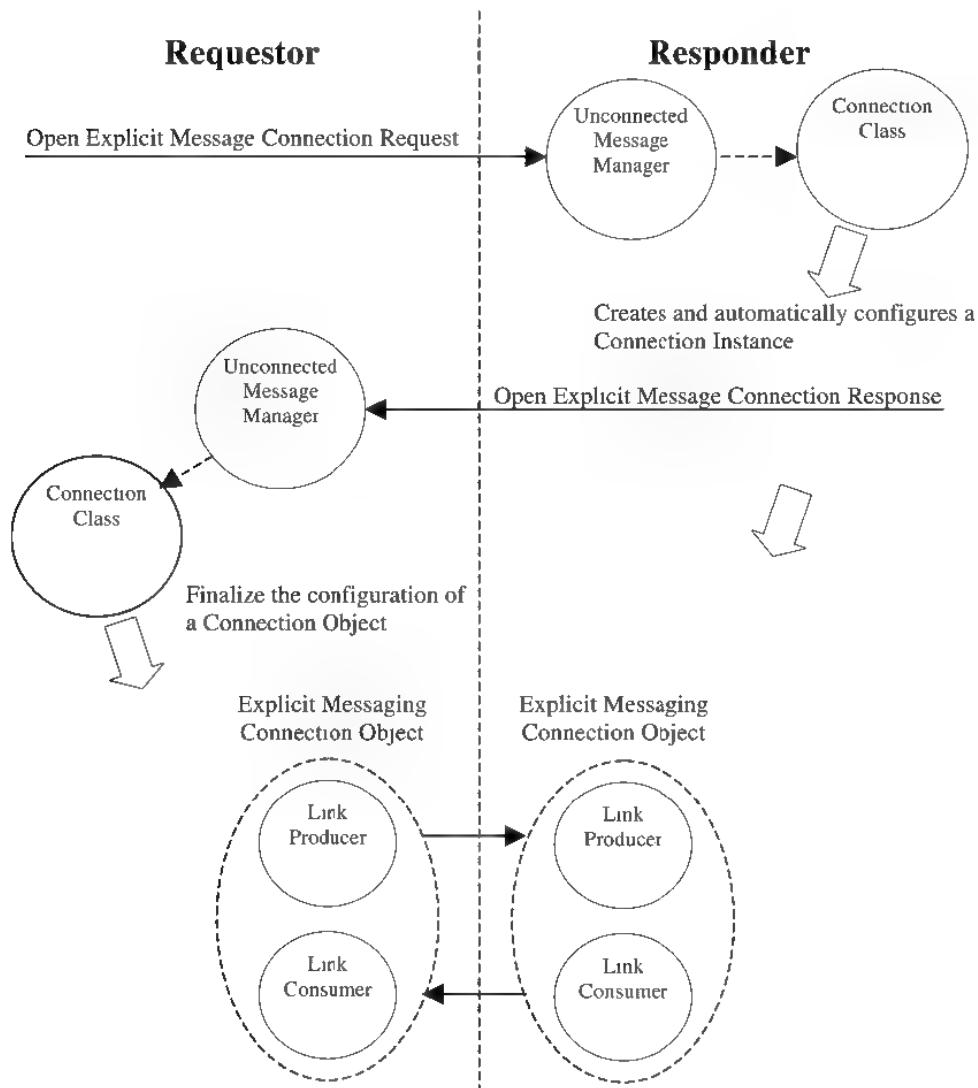
Devices that support the UCMM (UCMM capable devices) must also screen for the Duplicate MAC ID Check Message (section 2-3.3) and the Connection ID(s) associated with any other established connection(s).

These screening requirements will, in all likelihood, cause an implementation to utilize a Mask-and-Match CAN chip screener that is capable of receiving ALL Group 3 Messages. With this in mind, it is possible for implementations that support the UCMM to receive a large number of *message received* indications that must be screened in software. Resource restrictions typically associated with lower level devices may prohibit this level of software screening. The *Predefined Master/Slave Connection Set* previewed in section 2-5 and formally defined in Chapter 3, section 3-6 enables the implementation of a device that does not support the UCMM.

**Important:** Every effort should be made to support the UCMM. As outlined in Chapter 3, section 3-15, devices that do not support the UCMM place an additional burden on network resources.

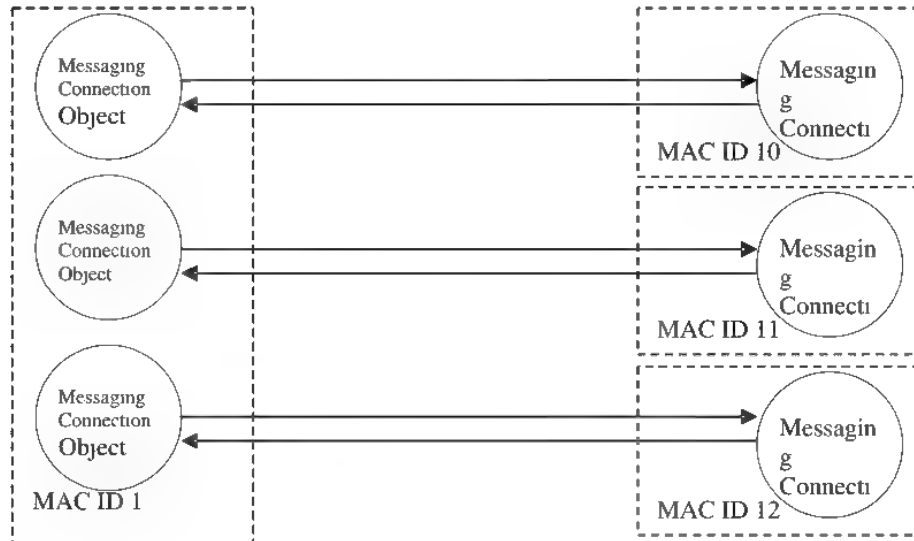
Figure 2-4.2 illustrates the steps involved in establishing a Messaging Connection.

**Figure 2-4.2 Establishing an Explicit Messaging Connection**



Explicit Messaging Connections are unconditionally *point-to-point*. Point-to-point connections exist between two devices ONLY. The device that requests that the connection be opened (the originator) is one *end-point* of the connection, and the module that receives and responds to the request is the other *end-point*. See Figure 2-4.3.

**Figure 2-4.3 Point-to-Point Nature of Explicit Messaging Connections**



## 2-4.2 I/O Connections

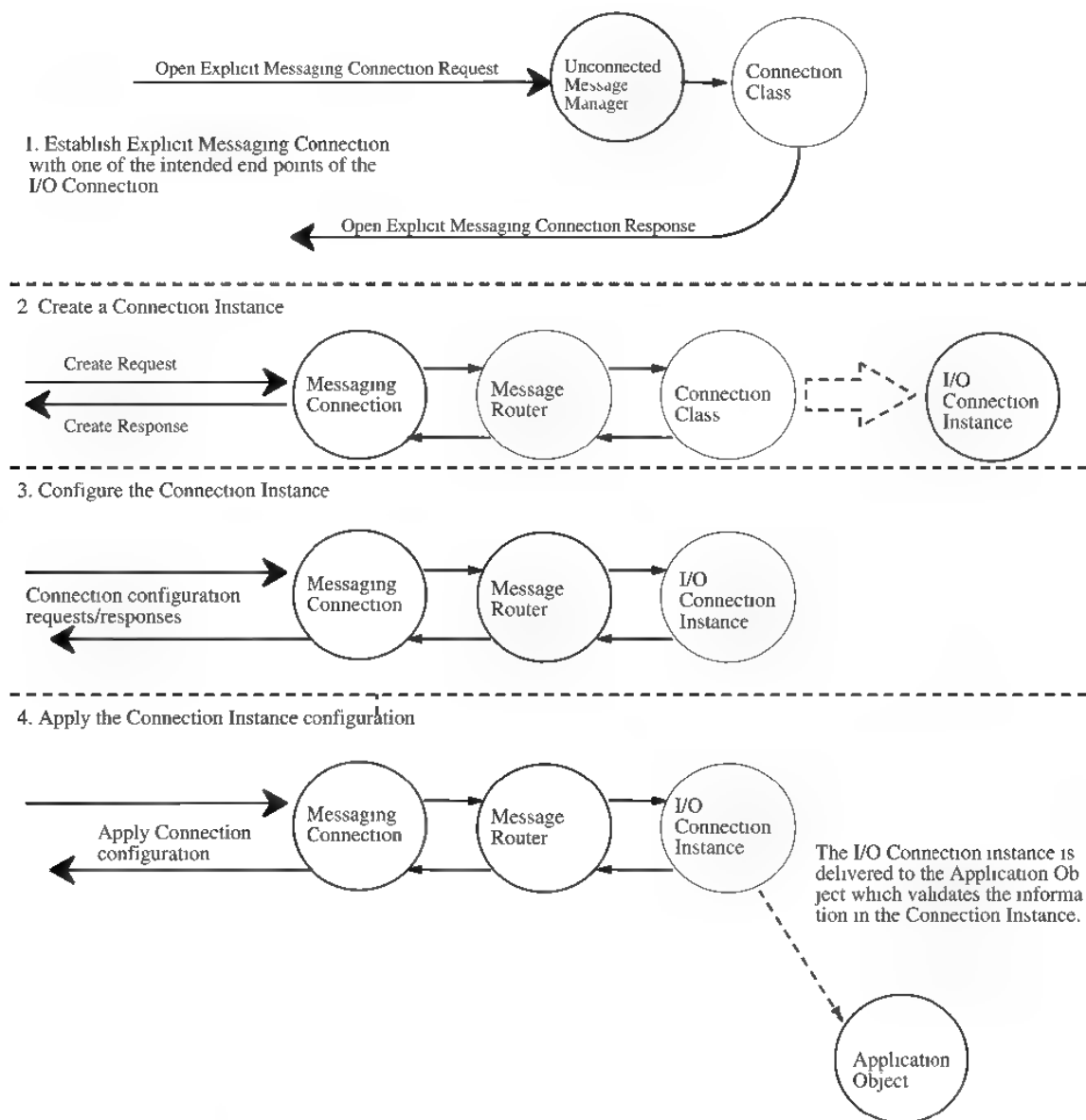
Dynamic I/O Connections are established by interfacing with the Connection Class across a previously established Explicit Messaging Connection. The following list presents the tasks necessary to dynamically establish an I/O connection:

1. Establish an Explicit Messaging Connection with one of the intended end points of the I/O Connection
2. Create an I/O Connection Object by sending a Create Request to the DeviceNet Connection Class
3. Configure the Connection instance
4. Apply the configuration performed at the I/O Connection Object. Doing so results in the instantiation of the components necessary to service the I/O Connection.
5. Repeat this process within the other end point.

**Important:** DeviceNet does NOT require support for the **dynamic** establishment of I/O Connections. Certain products may perform many of these steps automatically upon the detection of a particular event. The configuration of each and every part of an I/O Connection may not have to be performed in the dynamic fashion illustrated in Figure 2-4.4.

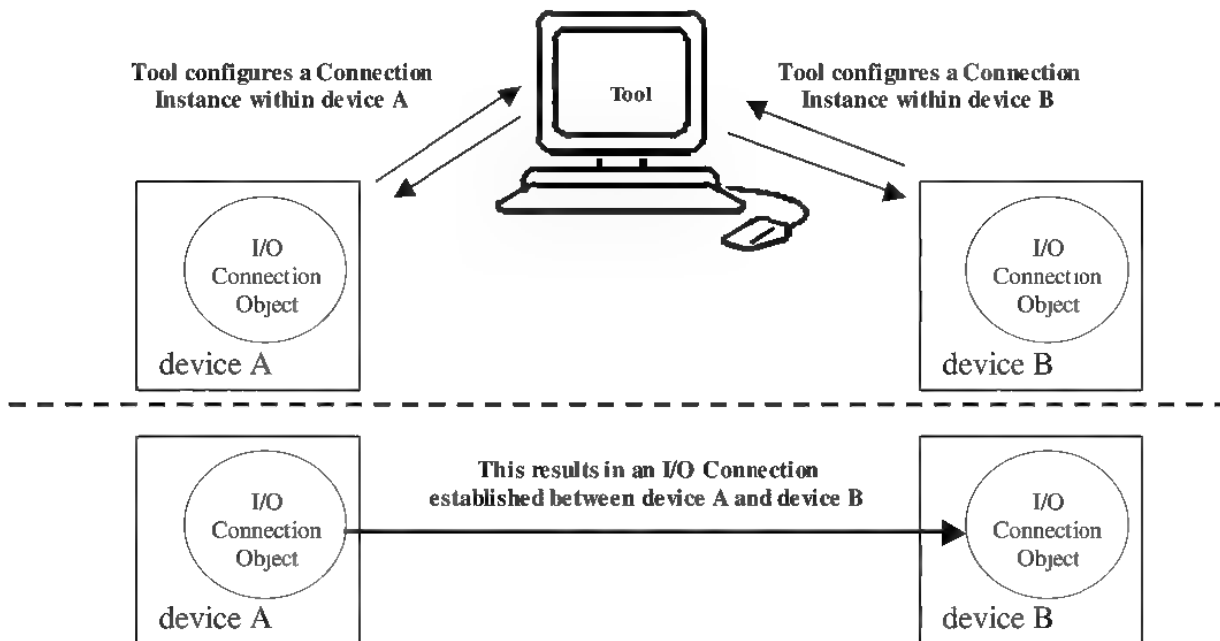


Figure 2-4.4 Dynamic Establishment of an I/O Connection



The dynamic process facilitates the establishment of a variety of I/O Connections. This specification does not dictate any rules associated with *who* may perform Connection configuration. For example, a tool could interface with two separate devices and create an I/O Connection between them. See Figure 2-4.5.

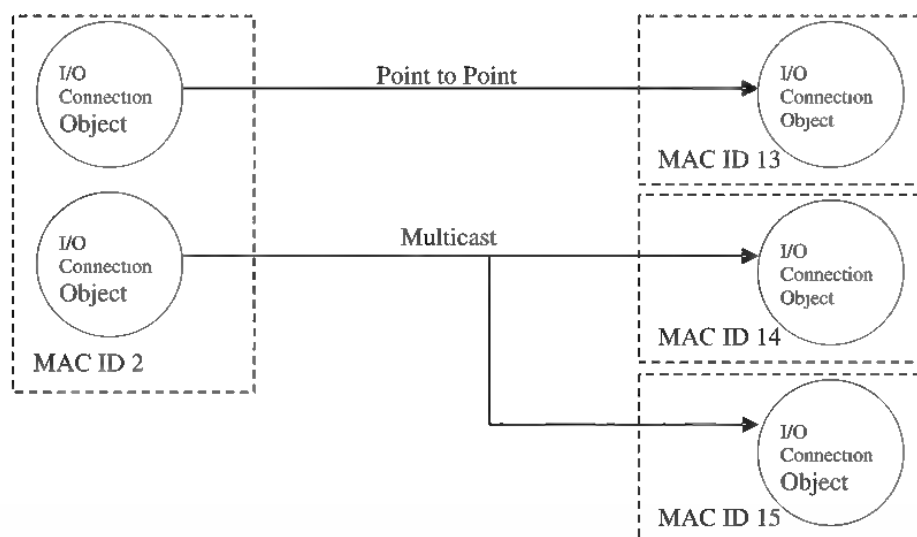
Figure 2-4.5 Tool Interface with Devices to Create Connection



The tool uses various Explicit Messaging Services to create and configure the I/O Connection Objects within the end points.

I/O connections can be either point-to-point or *multicast*. Multicast connections allow a single transmission to be heard by many nodes. See Figure 2-4.6.

Figure 2-4.6 Point-to-Point or Multicast Nature of I/O Connections



## 2-5 Predefined Master/Slave Connection Set

The preceding sections have presented the “*general model*” rules for establishing connections between devices. The general model calls for the utilization of an Explicit Messaging Connection to configure communication parameters within each connection end-point.

This specification uses the general model as a basis for the definition of a set of connections which facilitate communications typically seen in a Master/Slave relationship. These Connections are referred to collectively as the *Predefined Master/Slave Connection Set*.

Many of the steps involved in the creation and configuration of an Application to Application connection have been removed within the Predefined Master/Slave Connection Set. This, in turn, presents the means by which a communication environment can be established using less network and device resources.

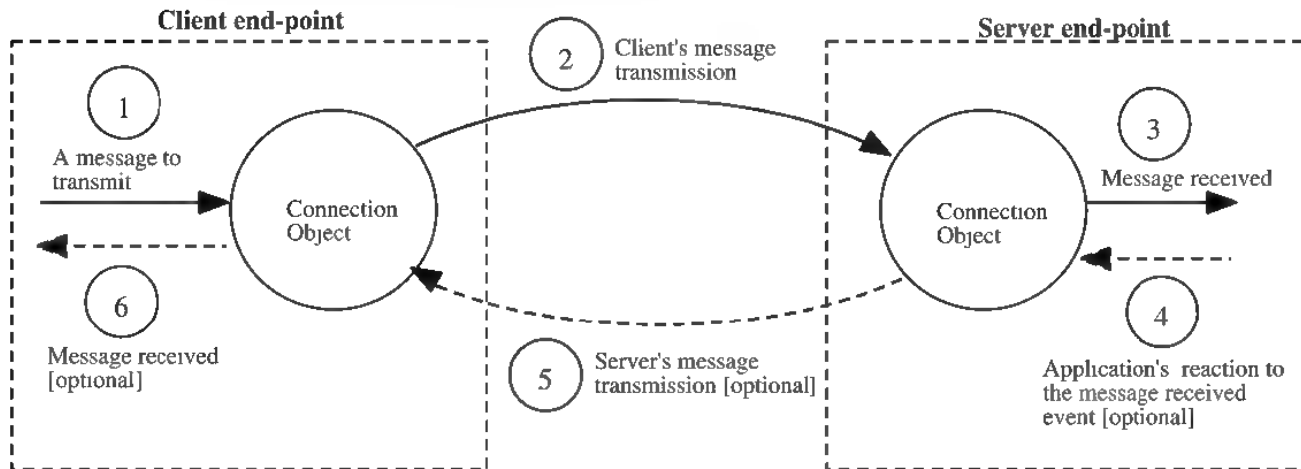
As mentioned in section 2-4.1, the mechanisms defined to manage the Predefined Master/Slave Connection Set enable an implementation that does not support the UCM.

Chapter 3 presents the characteristics of the Connections involved in the Predefined Master/Slave Connection Set and defines the means by which their use is managed.

## 2-6 Client and Server Connection End-points

The terms *Client* and *Server* are used throughout this document when discussing the behavior associated with a connection end-point. A Client end-point and Server end-point(s) are associated with both Explicit Messaging and I/O Connections. The *Client* is the module that originates a transmission, and the *Server* is the module that reacts to that transmission. The Server’s reaction may cause it to return a message to the Client.

Figure 2-6.1 Client Server Connection End Points



## 2-7 DeviceNet Messaging Protocol

The following sections describe the protocol information located within the CAN Data Field for both Explicit and I/O messages. See Chapter 9, section 9-5 for a description of how DeviceNet uses CAN.

### 2-7.1 Explicit Messaging

This section describes the Explicit Messaging Protocol and presents details associated with the dynamic establishment of Explicit Messaging Connections. An Explicit Message uses the Data Field of a CAN frame to carry DeviceNet defined information.

Figure 2-7.1 Explicit Message CAN Data Field Use



Figure 2-7.2 provides the format of the CAN Data Field associated with Explicit Messages.

Figure 2-7.2 Explicit Message Data Field Format

Byte Offset	Contents							
	7	6	5	3	5	2	1	0
0	Message header							
1	Message Body							
..								
7								

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Message header							
1	Fragmentation Protocol							
2	Message Body							
..								
7								

The data field of a transmission that contains the complete Explicit Message includes:

- A Message Header
- The entire Message Body

If the Explicit Message is greater than eight (8) bytes in length, it must be transmitted on DeviceNet in a *fragmented* manner. The fragmentation/re-assembly function is provided by the Connection Object. A fragmented piece of an Explicit Message includes:

- A Message Header
- The Fragmentation Protocol (see section 2-9, Fragmentation/Reassembly)
- A Message Body Fragment

### 2-7.1.1 Message Header

The **Message Header** is specified within byte offset zero (0) in the CAN Data Field of an Explicit Message and is formatted as follows:

**Figure 2-7.3 Explicit Message Header Format**

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Frag	XID	MAC ID					

#### Message Header Contents:

**Frag** (Fragment Bit) - The field that indicates whether or not this transmission is a fragmented piece of an Explicit Message. The following values are defined:

**Table 2-7.1 Frag Bit Values**

Value	Meaning
0	<i>Non-fragmented.</i> This transmission contains the complete Explicit Message. Next byte contains the Service field.
1	<i>Fragmented.</i> This transmission does not contain the complete Explicit Message. Next byte contains the Fragmentation Protocol.

**XID (Transaction ID)** - This field is utilized by an application to match a response with the associated request. This field is simply echoed by the Server in a response message. The Server module **DOES NOT** utilize this field to perform any type of duplicate message detection logic. This field contains a don't care value when a Client sends an Explicit Message for which a response is not expected.

**MAC ID** - Contains either the Source or Destination MAC ID. Refer to Table 2-7.2 MAC ID Within The Message Header to determine which MAC ID (source or destination) should be specified in this field.

**Table 2-7.2 MAC ID Within The Message Header**

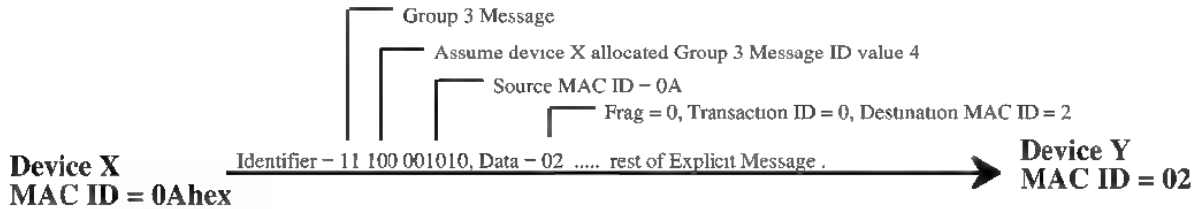
If:	Then:
The Destination MAC ID is specified in the Connection ID (CAN Identifier Field)	The Source MAC ID is specified in the MAC ID portion of the Message Header
The Source MAC ID is specified in the Connection ID (CAN Identifier Field)	The Destination MAC ID is specified in the MAC ID portion of the Message Header.

When an Explicit Message is received, the MAC ID field within the Message Header is examined. If the Destination MAC ID is specified in the Connection ID, then the Source MAC ID of the other end-point must be specified in the Message Header. If the Source MAC ID is specified in the Connection ID, then the receiving module's MAC ID must be specified in the Message Header. If either of these checks fail, then the message is discarded.

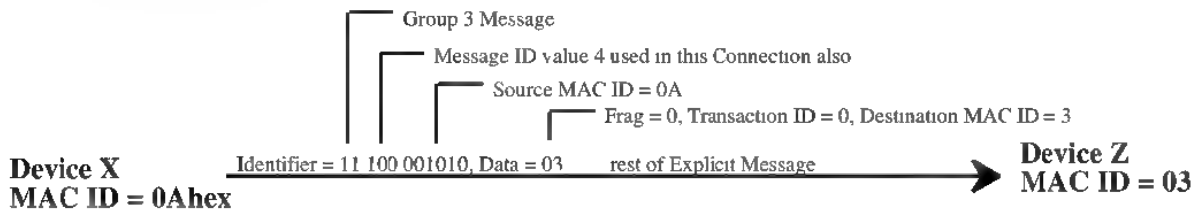
A side-effect of the examination of the MAC ID within the Message Header is the ability to use the same Connection ID to denote traffic with different **Explicit Messaging Connections to different devices.** This can only be done when the Explicit Messaging Connection ID contains the source device's MAC ID. This is due to the fact that the Destination MAC ID will then be specified in the Message Header and it is also used during reception processing. For example;

**Figure 2-7.4 Explicit Messaging Connection ID Re-Use**

Assume a device whose MAC ID is 0Ahex has established an Explicit Messaging Connection across Message Group 3 with a device whose MAC ID is 02



Since the Destination MAC ID is specified in the CAN Data Field, device X can use Message ID 5 in another Explicit Messaging Connection with another device while it is still being used in the first connection



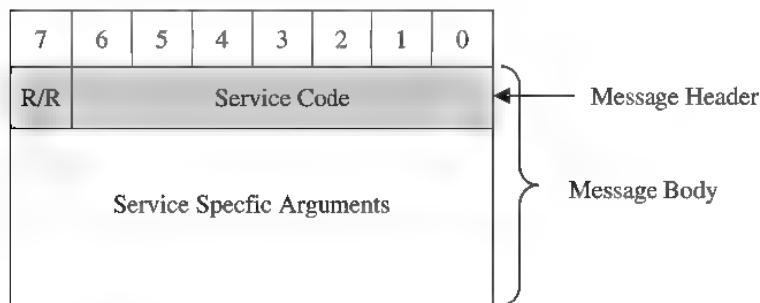
This increases the number of Explicit Messaging Connections a device can establish. The drawback to this approach is that a device will experience an unnecessary network interrupt when an Explicit Message destined to a different device is transmitted. For example; Referring to Figure 2-7.4, Device Z will experience a network interrupt whenever Device X sends a message to Device Y.

### 2-7.1.2 Message Body

The **Message Body** contains a *Service Field* and *Service Specific Arguments*.

The first argument specified within the Message Body is the **Service Field**, which identifies the particular request or response being delivered. Figure 2-7.5 illustrates the format of the Service Field.

**Figure 2-7.5 Service Field Format**



#### Service Field Contents:

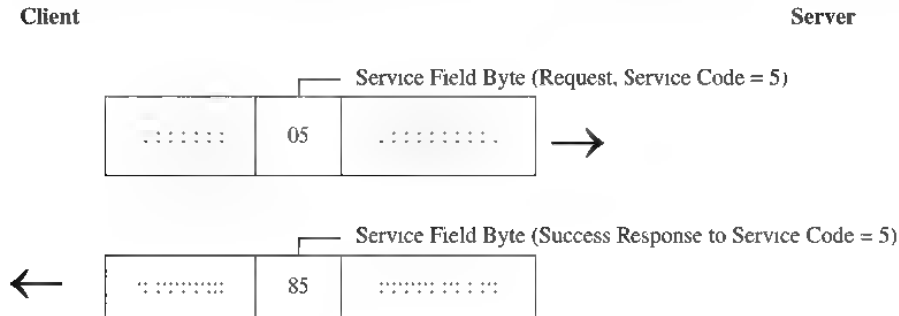
- **Service Code** - the value specified within the least significant seven (7) bits of the Service Field byte that indicates the type of service being transmitted.
- **R/R** - the most significant bit in the Service Field. Its value determines whether a message is a request or a response. See Table 2-7.3.

Table 2-7.3 R/R Bit Specification

If the value of the R/R Field is:	Then:
zero (0) (bit cleared)	the message is a Request.
one (1) (bit set)	the message is a Response.

When the Server generates a response indicating that a request was successfully serviced, the *Service Field* contains the request's Service Code with the Request/Response Flag set to one (1).

Figure 2-7.6 Service Field Byte for Explicit Request and Response



DeviceNet defines a set of **common services**. DeviceNet *common* services are the Open set whose parameters and required behaviors are defined in this document (see Volume 1, Appendix A, Explicit Messaging Services).

Following the *Service Field* within the Message Body is information specific to the particular type of service being transmitted.

## 2-7.2 UCMM Services

The Unconnected Message Manager (UCMM) provides for the dynamic establishment of Explicit Messaging Connections. This section presents a detailed description the Service Specific arguments associated with the Open Explicit Messaging Connection and Close Connection services provided by the UCMM. This serves as a prefix to describing the Connection Based Messaging protocol presented in section 2-7.3.

The UCMM processes two services, which manage the allocation and deallocation of Explicit Messaging Connections:

- Open Explicit Messaging Connection - Service Code = 4B<sub>hex</sub>. Used to establish an Explicit Messaging Connection.
- Close Connection - Service Code = 4C<sub>hex</sub>. Used to delete a Connection Object and deallocate all associated resources.

These services are accessed using the Unconnected Explicit Request and Response CAN Identifier Fields defined in section 2-2.3. When processing an Unconnected Explicit Request the UCMM may need to return an error indication to the requester and, as such, the Error Response Explicit Message can be transmitted with the Unconnected Explicit Response CAN Identifier.

### 2-7.2.1 Open Explicit Messaging Connection Request

This service requests the establishment of a logical connection between two modules across which Explicit Messages will be transmitted. This service is transmitted as an *Unconnected Request* Message (Message Group 3, Message ID 6).

**Figure 2-7.7 Open Explicit Messaging Connection Request Format**

Contents									
Byte Offset	7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID						} Message Header
1	R/R [0]	Service Code [4B]							
2	Reserved (all bits=0)				Requested Message Body Format				} Message Body
3	Group Select				Source Message ID				

#### 2-7.2.1.1 Open Explicit Message Connection Request Data Frame Contents

**Frag (0)/Transaction ID/MAC ID** - See section 2-7.1.1, Message Header. Note that the Destination MAC ID is always specified in the Message Header associated with an Open Explicit Messaging Connection Request/Response.

**R/R Bit (0)** - Indicates this is a request message.

**Service Code (4B<sub>hex</sub>)** - Identifies this as an Open Explicit Messaging Connection service.

**Reserved Bits** - Use is to be developed. These bits currently are ignored by the receiver and should be set to zero by the transmitter.

**Requested Message Body Format**- The field used by the Client to request a particular Message Body format for subsequent Explicit Messages transmitted over this connection. A server shall support at least one of the DeviceNet Message Body Formats (values 0 – 3).

**Important:** The Server module responding to this Open Explicit Messaging Request defines the actual message body format to be used over this connection. See Table 2-7.4 for Message Body Format values. Servers can do one of the following:

- Refuse the request and return a supported format in the Open Explicit Messaging Connection Response. The returned format shall be one of the DeviceNet Message Body Formats (values 0 – 3).
- Accept this request by echoing the same value in the Open Explicit Messaging Connection Response.



Table 2-7.4 Message Body Format Values

Value	Meaning
0	DeviceNet 8/8. Class ID = 8 bit integer, Instance ID = 8 bit integer <sup>1</sup>
1	DeviceNet 8/16. Class ID = 8 bit integer, Instance ID = 16 bit integer <sup>1</sup>
2	DeviceNet 16/16. Class ID = 16 bit integer Instance ID = 16 bit integer <sup>1</sup>
3	DeviceNet 16/8. Class ID = 16 bit integer Instance ID = 8 bit integer <sup>1</sup>
4	CIP Path. The addressing size is variable and is provided as a Packed EPATH on each request <sup>1 2</sup>
5 - F	Reserved by DeviceNet

- 1 Messages transmitted across this connection are formatted as described in section 2-7.3, Connection Based Explicit Messaging.
- 2 The CIP Path is specified as an 8-bit path length (USINT) followed by the path (Packed EPATH). See Volume 1, Appendix C, Data Management for information on EPATH. The server shall support, at a minimum, the most efficient encoding for any logical segment within the EPATH (for example, 8-bit class encoding for class code 0x01).

**Group Select** - The field that indicates the Message Group across which messages associated with this connection are to be exchanged. The following values are defined:

Table 2-7.5 Group Select Values

Value	Meaning
0	Message Group 1
1	Message Group 2 <sup>1</sup>
2	Reserved
3	Message Group 3
4 - E	Reserved by DeviceNet
F	Reserved for node ping <sup>2</sup>

- 1 By definition, the Message Group 2 Identifier allows for specification of either the Source or Destination MAC ID. For Explicit Messaging Connections established across Message Group 2, the Client places the MAC ID of the Server (Destination MAC ID) in the Connection ID when transmitting messages across this connection. The Server places its own MAC ID (Source MAC ID) in the Connection ID when transmitting messages across this connection. This process requires the Server to allocate two separate Message IDs from its Group 2 *pool* to ensure that itself and the Client do not have the ability of transmitting the identical bit pattern in the CAN Identifier Field
- 2 This group select value can be used to force a UCMM capable target node to respond without creating any resources within the node. When a request with this value is received, the target shall respond with a General Error of Invalid Parameter (0x20) and an Additional Error Code of 0x01.

**Important:** The Client selects the Message Group across which the transmissions associated with this Explicit Messaging Connection will take place. If the Server cannot satisfy the request, then it must reject the request and return an Error Response.

**Source Message ID-** The use of this field depends on the value within the Group Select field. Refer to Table 2-7.5.

**Table 2-7.6 Source Message ID in Open Explicit Messaging Connection Request**

If Group Select equals:	Then the Source Message ID:
0 or 3	Specifies the Message ID the Client has allocated from its Group 1 or 3 Message ID <i>pool</i> . The Client will use this Message ID in combination with its own MAC ID (Source MAC ID) to generate the Connection ID specified when it subsequently transmits a message across this Connection. <sup>1</sup>
1	Is ignored/set to the value zero (0). <sup>2</sup>

- 1 The Client places this value within the Message ID component of the Message Group 1 or 3 Identifier. This gives the Server enough information to configure its associated Link Consumer and CAN Chip to receive transmissions performed by the Client. See the examples that follow the argument descriptions.
- 2 Explicit Messaging Connections established across Message Group 2 require the Server to allocate two Message IDs from its Group 2 *pool* and return them in the Open Explicit Messaging Connection Response Message. The Client will use one of these Message IDs to generate the Connection ID it specifies when transmitting a message across this connection. The other will be used by the Server to generate the Connection ID it specifies when transmitting a message across this connection. This provides sufficient information for the Client and Server to configure their respective Link Producers/Consumers and CAN chips. See the examples that follow the argument descriptions for a detailed description.

### 2-7.2.1.2 Service Procedure

The UCMM within the Server validates the Open Explicit Messaging Connection Request arguments. If they are valid, the UCMM invokes the Create service of the Connection Class to obtain a Connection Object Instance (see section CIP Common, Chapter 3, section 3-4.5). The resulting Connection Object is automatically configured to be an Explicit Messaging Connection Object.

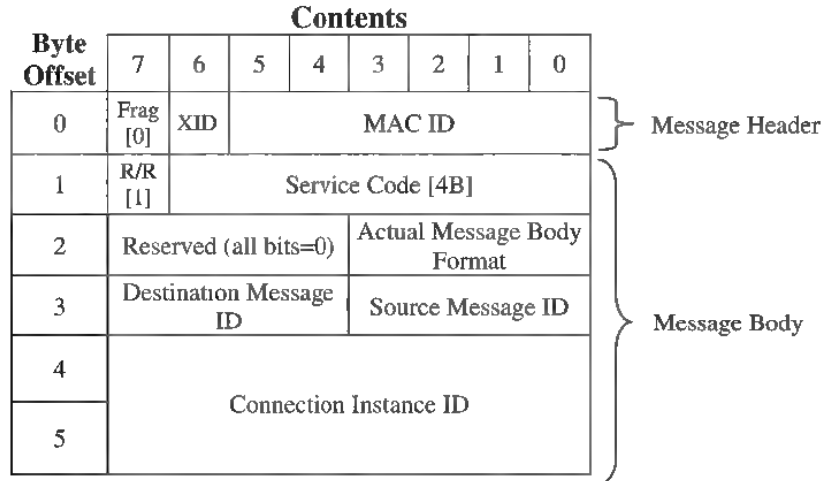
If the Server supports multiple Message Body Formats and the Client has requested one of those formats, then the Server honors that request by echoing the *Requested Message Body Format* in the Open Explicit Response message. If the Server does not support multiple Message Body Formats, then the Server simply specifies its default format in the Open Explicit Messaging Connection Response.

If no errors are detected, then an Open Explicit Messaging Connection Response is returned. If an error is detected, then an Error Response Message is returned.

### 2-7.2.2 Open Explicit Messaging Connection Success Response

This service is used to respond successfully to an Open Explicit Messaging Connection Request Message.

Figure 2-7.8 Open Explicit Messaging Connection Response Format



#### 2-7.2.2.1 Open Explicit Message Connection Response Data Frame Contents

**Frag (0)/Transaction ID/MAC ID** - See section 2-7.1.1, Message Header. Note that the Destination MAC ID is always specified in the Message Header associated with an Open Explicit Messaging Connection Request/Response.

**R/R Bit (1)** - Indicates that this is a response message.

**Service Code (4B<sub>hex</sub>)** - Identifies this as an Open Explicit Messaging Connection service.

**Reserved Bits** - Use is to be developed. These bits are currently ignored by the receiver and should be set to zero by the transmitter.

**Actual Message Body Format** - The field used by the Server to define the format of the Message Body associated with subsequent Explicit Messages transmitted over this connection. When the Open Explicit Messaging Connection Request was transmitted, the Client requested a particular format within the *Requested Message Body Format* argument. Refer to Table 2-7.4 for Message Body Format values.

**Destination Message ID** - The use of this field depends on the Message Group across which the Client requested this connection take place.

Table 2-7.7 Destination Message ID-Open Explicit Messaging Connection Response

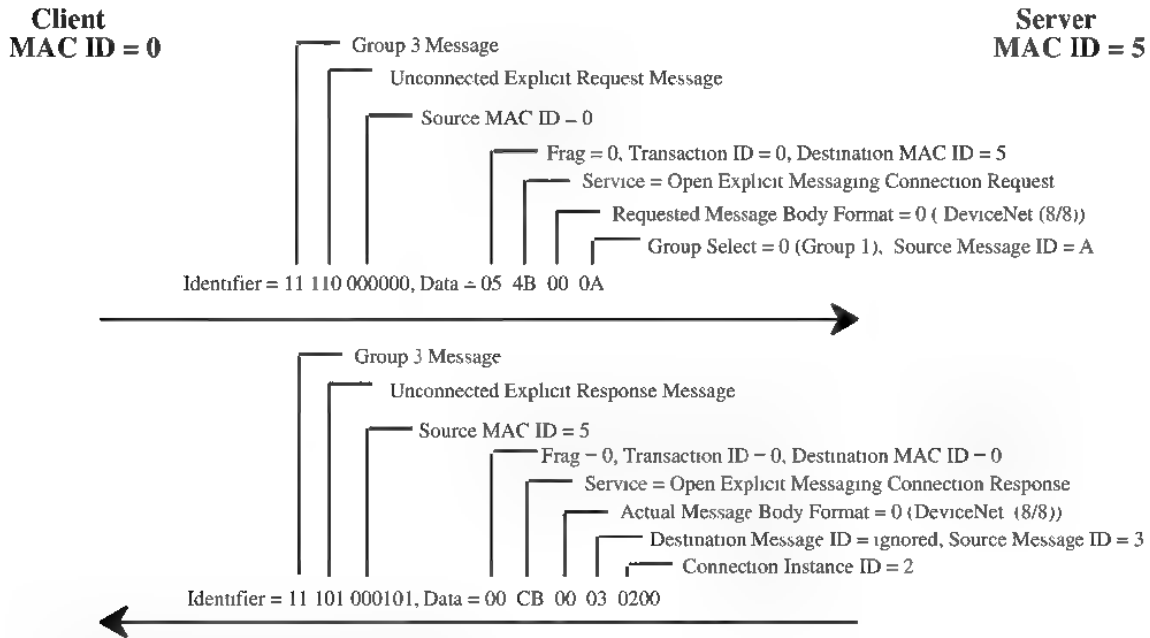
If Group Select within the Open Request was set to:	Then the Destination Message ID in the Open Response:
0 or 3	Is ignored and should be set to the value zero (0).
1	Is used by the Client in combination with the Server's MAC ID (Destination MAC ID) to generate the Connection ID it specifies when transmitting across this connection. The Server has allocated this value from its Group 2 Message ID pool.

**Source Message ID** - The Message ID value the Server has allocated. The Server allocates a Message ID from its Group 1, 2, or 3 Message ID pool that will be used in conjunction with its own MAC ID (Source MAC ID) to generate the Connection ID that is specified when it transmits a message across this Connection. This gives the Client enough information to configure its associated Link Consumer and CAN Chip to receive transmissions performed by the Server.

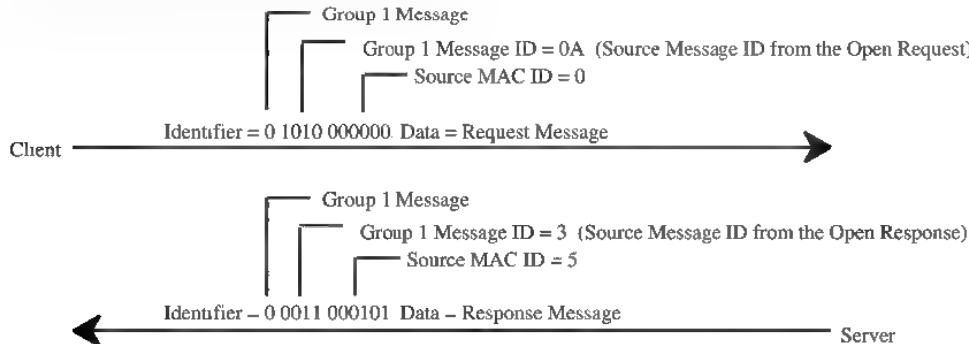
**Connection Instance ID** - The Server instantiates an Explicit Messaging Connection Object when it successfully services an Open Request. This field holds the Instance ID value assigned to that Explicit Messaging Connection Object. The Server returns this value, which can subsequently be used by the Client when it wants to close the connection. The Connection Instance ID field is specified within a 16-bit integer field (UINT) in the Open Explicit Messaging Connection Response.

### 2-7.2.3 Open Explicit Messaging Connection Protocol Examples

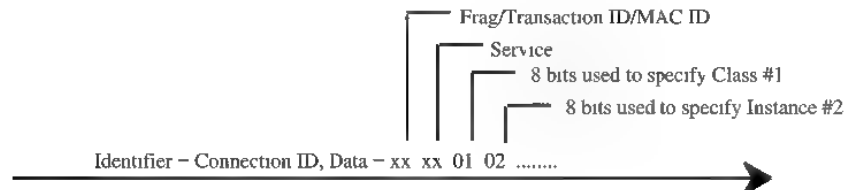
The three examples that follow illustrate an Open Explicit Messaging Connection Request/Response transaction.

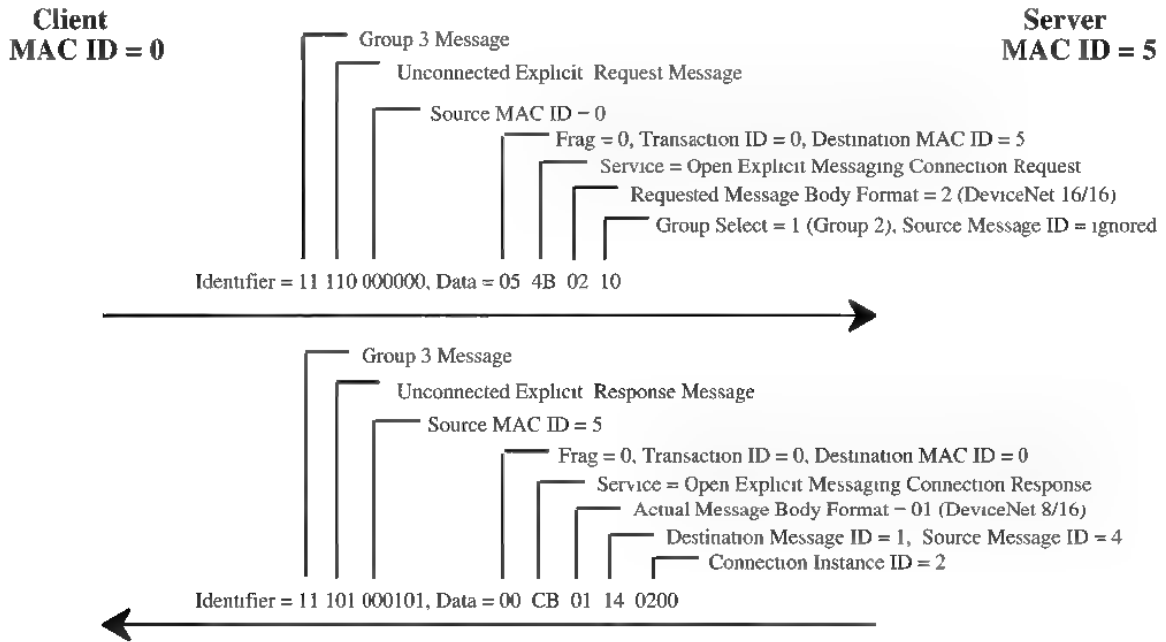


Note that the Server accepted the Requested Message Body Format by echoing the value in the Actual Message Body Format field. Subsequent transmissions from the Client and Server associated with this connection would use the following Connection IDs.

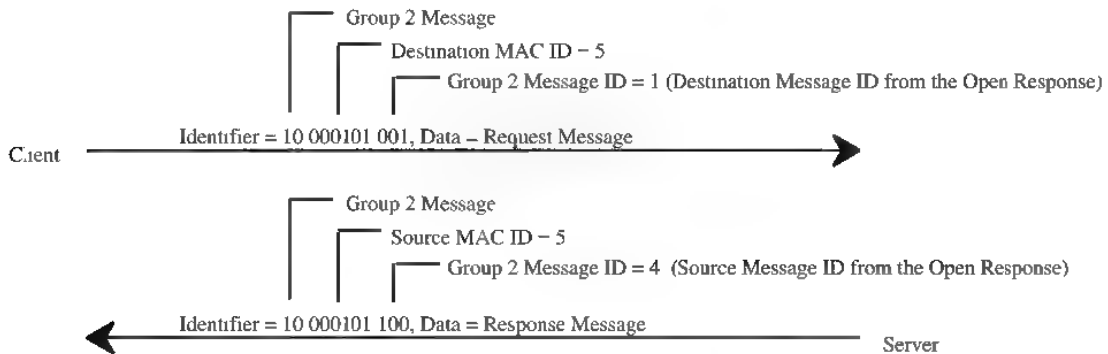


A request message to access Class 1/Instance 2 transmitted across this connection would specify that information as follows.

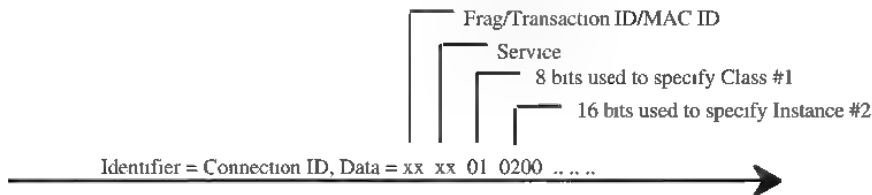


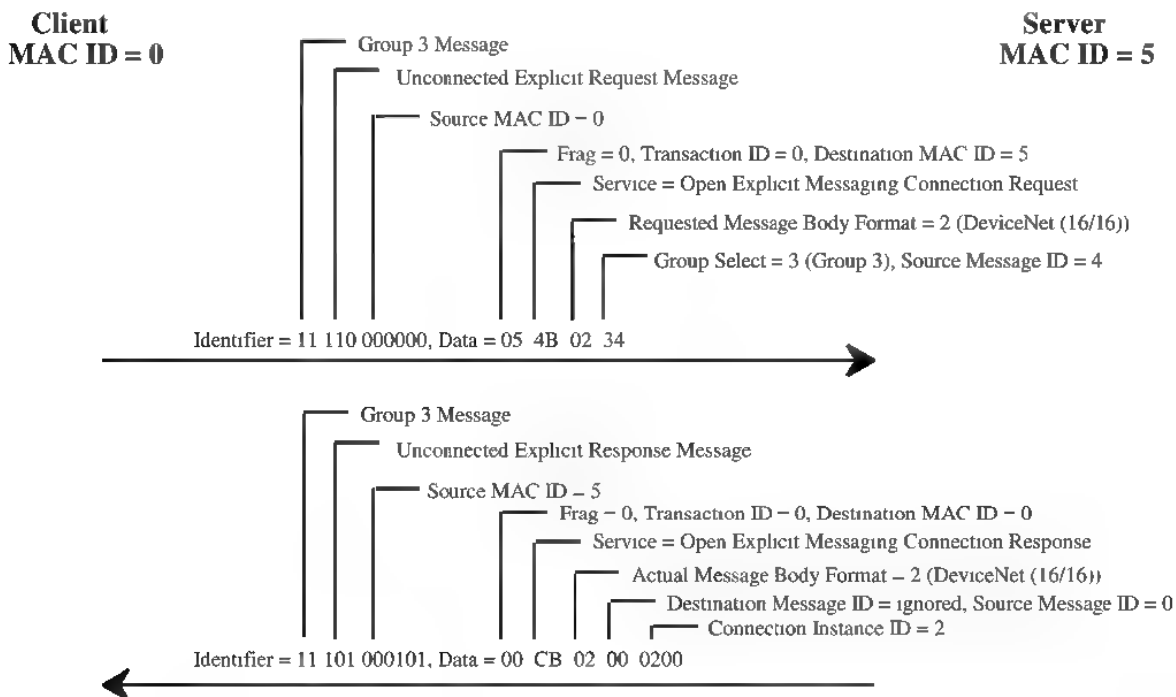


Note that the Server rejected the Requested Message Body Format and instead stated that the actual format will be DeviceNet (8/16). Subsequent transmissions from the Client and Server associated with this connection would use the following Connection IDs.

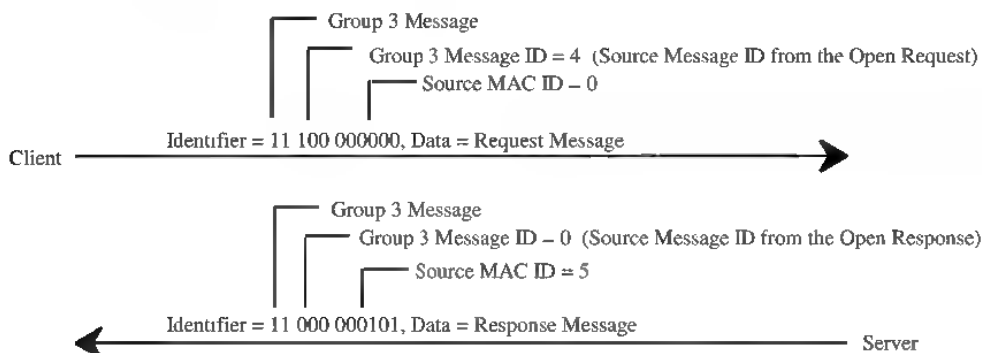


A request message to access Class 1/Instance 2 transmitted across this connection would specify that information as follows.

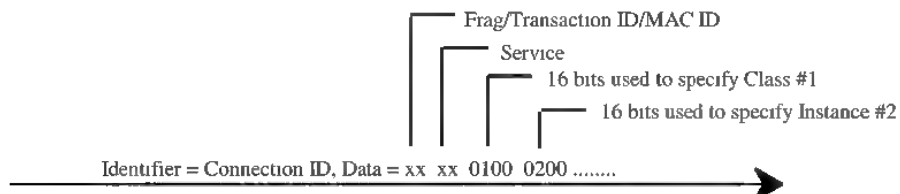


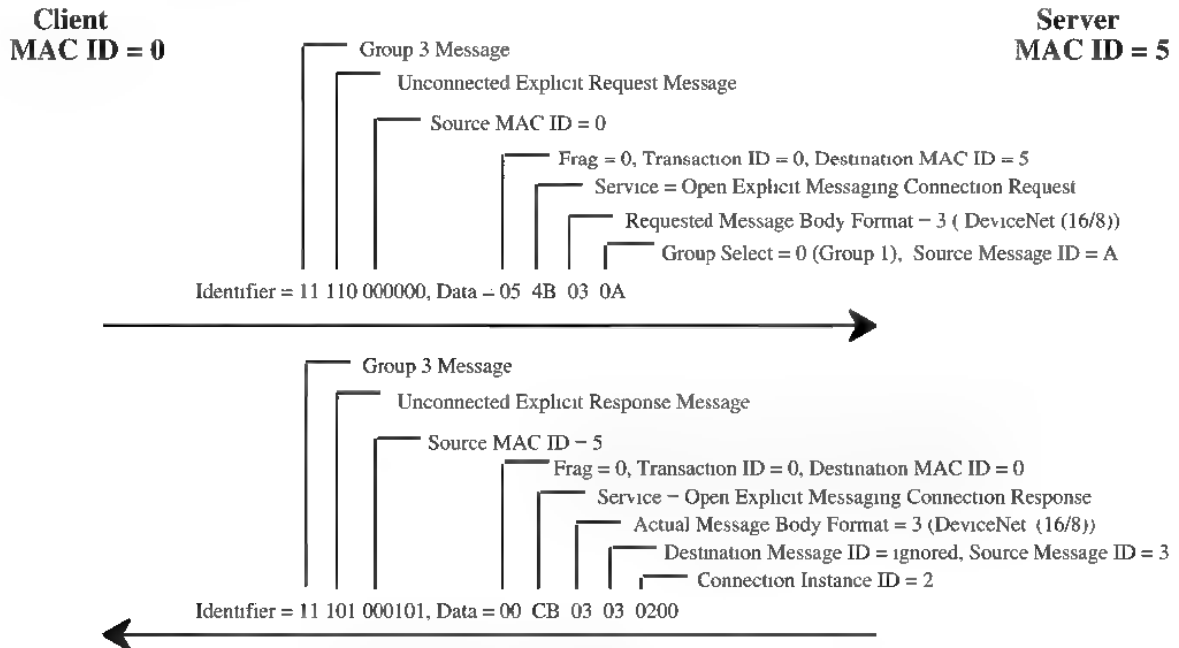


Note that the Server accepted the Requested Message Body Format by echoing the value in the Actual Message Body Format field. Subsequent transmissions from the Client and Server associated with this connection would use the following Connection IDs.

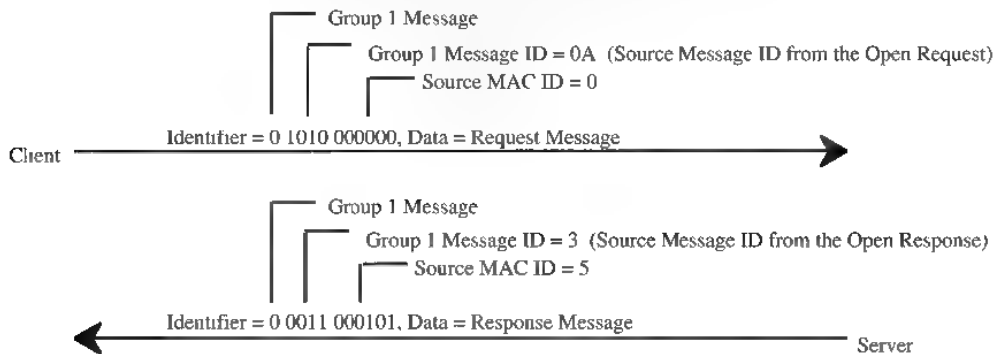


A request message to access Class 1/Instance 2 transmitted across this connection would specify the Object Address as follows:

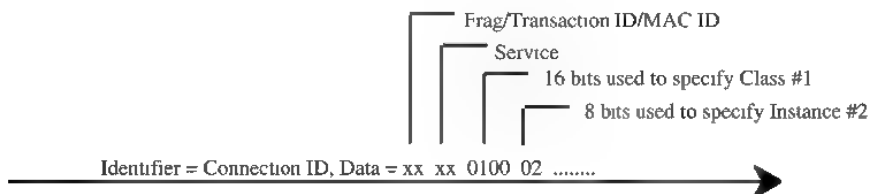




Note that the Server accepted the Requested Message Body Format by echoing the value in the Actual Message Body Format field. Subsequent transmissions from the Client and Server associated with this connection would use the following Connection IDs:

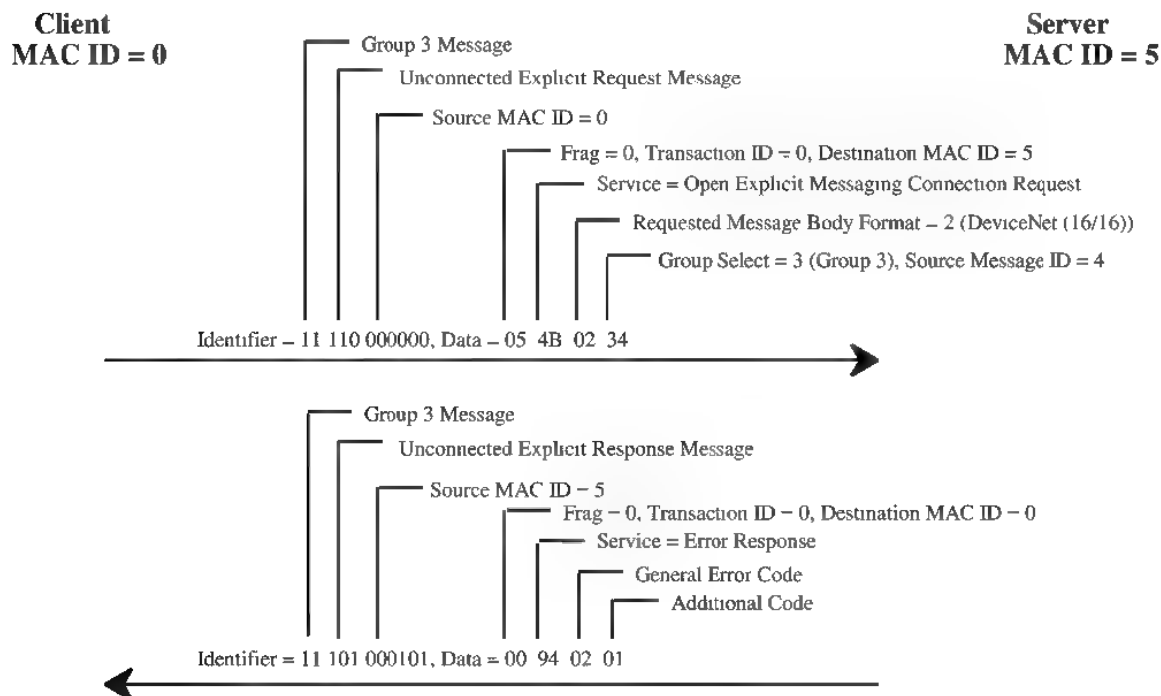


A request message to access Class 1/Instance 2 transmitted across this connection would specify that information as follows





The following example illustrates an Open Explicit Messaging Connection Request followed by an Error Response. A formal definition of the Error Response Message is presented in section 2-7.2.7.



#### 2-7.2.4 Close Connection Request

This service is used to terminate a Connection (either I/O or Messaging) within one of the end points. The reception of the Close message by the UCMM results in the invocation of the Connection Class's Delete service (see Volume 1, Chapter 3-4.5, Connection Object Instance Services). A Close Connection Request is transmitted as an Unconnected Request message (Message Group 3, Message ID 6).

**Important:** *Open* Explicit Messaging Connection Request/Response services establish **Explicit Messaging Connections ONLY**; whereas, the *Close* service terminates either type.

The Close Connection Request provides a means by which a connection can be deleted without having to establish an Explicit Messaging Connection. The transmission of the DeviceNet Common **Delete** service across an Explicit Messaging Connection to the Connection Class performs the same function, as does the Close service. The Delete service, however, can be transmitted only as a Connection Based Message (see section 2-7.3).

Figure 2-7.9 Close Connection Request Format

Byte Offset		Contents							
		7	6	5	4	3	2	1	0
0		Frag [0]	XID	MAC ID					} Message Header
1		R/R [0]	Service Code [4C]						
2		Connection Instance ID						Low byte	} Message Body
3								High byte	

Message Header

Message Body

##### 2-7.2.4.1 Close Connection Request Data Frame Contents

**Frag (0)/Transaction ID/MAC ID** - See section 2-7.1.1, Message Header. Note that the Destination MAC ID is always specified in the Message Header associated with a Close Connection Request/Response.

**R/R Bit (0)** - Indicates that this is a request message.

**Service Code (4C<sub>hex</sub>)** - Identifies this as a Close Connection service.

**Connection Instance ID** - The field that specifies the Connection instance to be deleted. Because the Close Connection Request message is transmitted as an Unconnected Message, the transmitter may not know any information about the *Message Body Format* associated with the intended recipient. As a result, the format for the Connection Instance ID within this message **is always specified as a 16-bit integer**.

##### Service Procedure:

The responder verifies that the specified Connection instance exists. If the Connection instance exists, and it can be deleted, then it is deleted. All resources associated with the Connection instance are freed. If the request is successfully serviced, then a Close Response is returned. If the request is not successful, an Error Response is returned.

## 2-7.2.5 Close Response

This service is used to respond successfully to a Close Request Message.

Figure 2-7.10 Close Connection Response Format

Byte Offset	Contents								
	7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID						} Message Header
1	R/R [1]	Service Code [4C]						} Message Body	

### 2-7.2.5.1 Close Connection Response Data Frame Contents

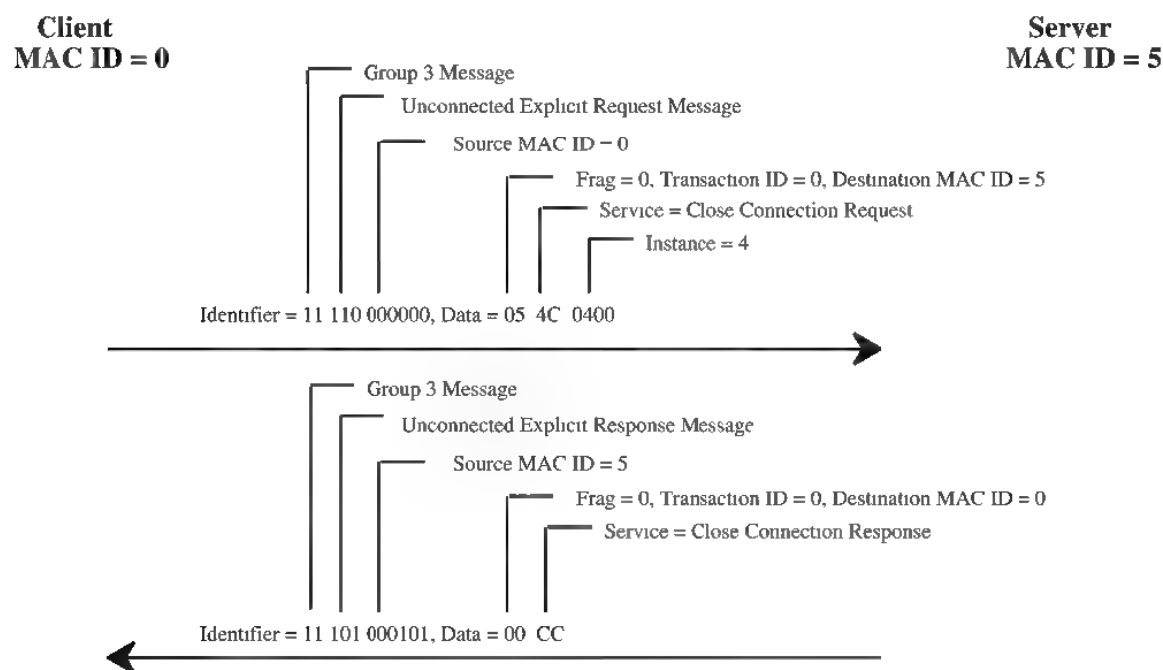
**Frag (0)/Transaction ID/MAC ID** - See section 2-7.1.1, Message Header.

**R/R Bit (1)** - Indicates that this is a response message. Note that the Destination MAC ID is always specified in the Message Header associated with a Close Connection Request/Response.

**Service Code (4C<sub>hex</sub>)** - Identifies this as a Close Connection service.

## 2-7.2.6 Close Connection Protocol Examples

The following example illustrates a Close Connection Request/Response transaction.



## 2-7.2.7 Error Response

See section 2-7.3.4 for a description of the format of an Error Response message. This section presents a standard set of UCMM related error conditions and the Error Code (both General Error Code and Additional Error Code) information to be used in an associated Error Response Message.

**Table 2-7.8 UCMM Error Conditions/Codes**

Error Condition	General Error Name	General Error Code (hex)	Additional Error Code (hex)
Service Code not Open or Close	Service not supported	08	FF
Group Select resource error	Resource unavailable	02	01
Group Select Out Of Range	Invalid parameter	20	01
Server out of Connections	Resource unavailable	02	02
Server out of Message IDs	Resource unavailable	02	03
Client Source Message ID Invalid	Invalid parameter	20	02
Duplicate Client Source Message ID	Resource unavailable	02	04
Connection Instance ID Invalid	Object does not exist	16	FF

Error condition descriptions:

- **Service Not Open or Close** - A service received across the UCMM port is not supported.
- **Group Select Resource Error** - The Group Select argument indicates utilizing a Message Group that is not supported by the device.
- **Group Select Out Of Range** - The Group Select field contains an invalid value.
- **Server Out of Connections** - The maximum number connections supported by this server has already been reached.
- **Server Out of Message ID's** - The server has allocated all Message ID's within the Message Group requested by the client.
- **Client Source Message ID Invalid** - The Source Message ID received with an Open Explicit Message Connection Request is invalid for the specified Message Group
- **Duplicate Client Source Message ID** - The Source Message ID and Source MAC ID received within an Open Explicit Messaging Connection Request are already in use for a Group 1 or Group 3 Explicit Messaging Connection. In this case, a request has been received from the same MAC ID specifying the same Group Select (1 or 3) and Source Message ID as an already existing Explicit Messaging Connection. For whatever reason, the Client has forgotten that it already had a Connection with that Server utilizing that particular set of Connection information (Group Select and Source Message ID).

It is valid for the Server to recognize this condition and send a successful UCMM Open Response. To send a successful UCMM Open Response when this condition is detected, the server must exhibit the following behavior:

1. Immediately delete the offending Messaging Connection
2. Process the UCMM Open Request in the normal fashion

Note that this condition cannot be detected when the Open Request specifies Message Group 2.

- **Connection Instance ID Invalid** - The Connection Instance ID received with the Close Connection Request does not exist.

### 2-7.3 Connection Based Explicit Messaging

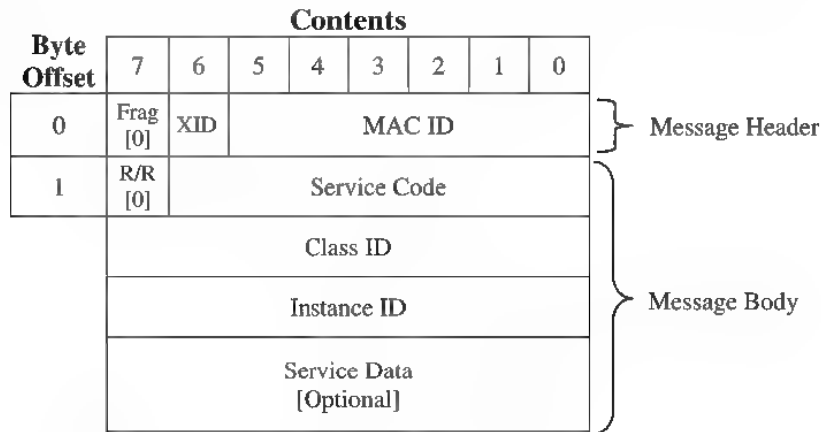
A *Connection Based Explicit Message* is a message transmitted over an Explicit Messaging Connection. This section defines the format for Connection Based Explicit Messages.

**Important:** A Connection Based Explicit Message must conform to the format described in this section. Conformance to this format is important when defining Object Class and/or Vendor Specific services. See Volume 1, Appendix A, Explicit Messaging Services, for information about service definitions.

#### 2-7.3.1 Explicit Request Data Frame (Message Body Format Values 0 – 3)

The illustration below depicts the format for the Message Body associated with a non-fragmented Explicit request for Message Body Format values 0 – 3:

Figure 2-7.11 Non-fragmented Explicit Request Message Body Format, Values 0 – 3



**Frag (0)/Transaction ID/MAC ID** - See section 2-7.1.1, Message Header.

**R/R Bit (0)** - Indicates that this is a request message.

**Service Code** - Defines the service being requested.

**Class ID** - Defines the object class towards which this request is directed. The Class ID is specified within either an 8- or 16-bit integer field based on the Actual Message Body Format value returned in the Open Explicit Messaging Connection Response.

**Instance ID** - Defines the particular instance within the object class towards which this request is directed. The Instance ID is specified within either an 8- or 16-bit integer field based on the Actual Message Body Format value returned in the Open Explicit Messaging Connection Response.

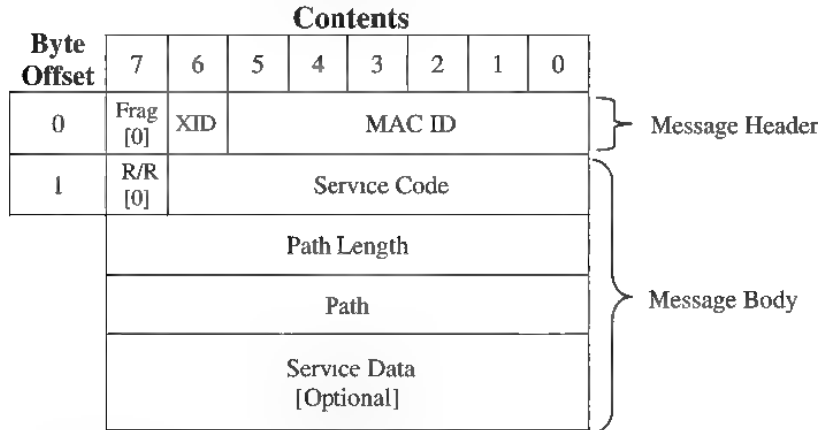
**DeviceNet reserves the value zero (0) to denote that the request is directed towards the class itself versus a specific instance within the class.**

**Service Data** - Carries request-specific data. Formats for the DeviceNet Common Services are presented in Volume 1, Appendix A, Explicit Messaging Services. Class/Object specific services must present a definition of the format of this field.

### 2-7.3.2 Explicit Request Data Frame Contents (Message Body Format value 4)

The illustration below depicts the format for the Message Body associated with a non-fragmented Explicit request for Message Body Format value 4:

Figure 2-7.12 Non-fragmented Explicit Request Message Body Format, Value 4



**Frag (0)/Transaction ID/MAC ID** – See section 2-7.1.1, Message Header.

**R/R Bit (0)** – Indicates that this is a request message.

**Service Code** – Defines the service being requested.

**Path Length** – This 8-bit value (USINT) provides the length of the message request path.

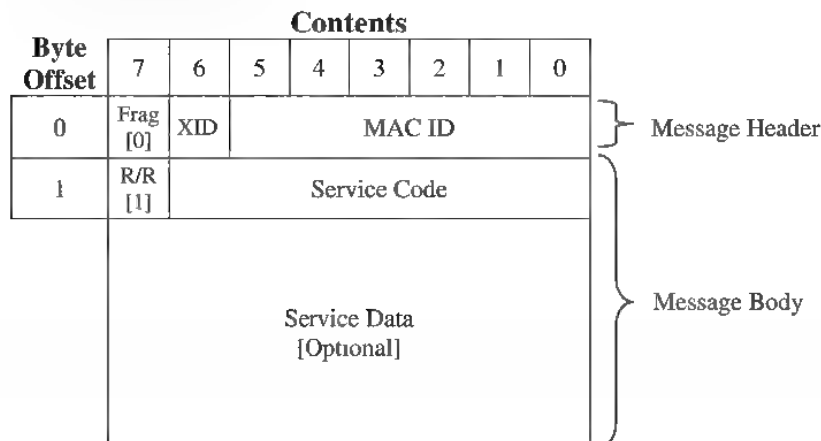
**Path** – The path of the message request (Packed EPATH).

**Service Data** – Carries request-specific data. Formats for the DeviceNet Common Services are presented in Volume 1, Appendix A, Explicit Messaging Services. Class/Object specific services must present a definition of the format of this field.

### 2-7.3.3 Success Response Explicit Message

The illustration below depicts the format for the Message Body associated with a non-fragmented positive/successful response:

Figure 2-7.13 Non-fragmented Successful Response Message Body Format



**Frag (0)/Transaction ID/MAC ID** - See section 2-7.1.1, Message Header.

**R/R Bit (1)** - Indicates that this is a response message.

**Service Code** - Contains the Service Code sent in the request message.

**Service Data** - Carries request specific data.

See Volume 1, Appendix A for examples of Connection Based Explicit Messaging.

#### 2-7.3.4 Error Response Explicit Message

The Error Response Explicit Message is returned when an error is encountered while attempting to service a previously received Explicit Request Message. The Error Response can be sent as either a Connection Based or Unconnected response message. If the request for which an Error Response is being returned was received across an Explicit Messaging Connection, then the Error Response is returned across that same Connection. If the request for which an Error Response is being returned was an Unconnected Explicit Request Message, then the Error Response is returned as an Unconnected Response Message. The following illustration depicts the format of an Error Response Explicit Message.

**Figure 2-7.14 Error Response Message**

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID					
1	R/R [1]	Service Code [14]						
2	General Error							
3	Additional							

**Frag (0)/Transaction ID/MAC ID** - See section 2-7.1.1, Message Header.

**R/R Bit (1)** - Indicates that this is a response message.

**Service Code (14<sub>hex</sub>)** - Identifies this as an Error Response.

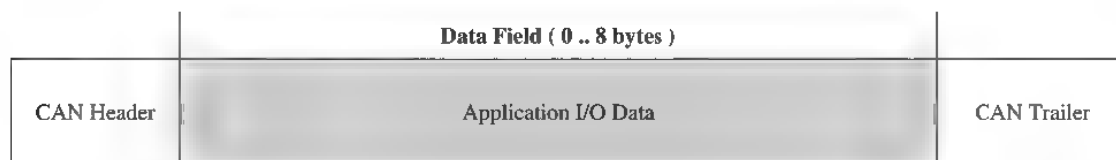
**General Error Code** - Identifies the encountered error. See Volume 1, Appendix B for a list of Status Codes.

**Additional Code** - Contains an object/service specific value that further describes the error condition. If the responding object has no additional information to specify, then the value FF<sub>hex</sub> is placed within this field.

## 2-8 Input/Output Messaging

Outside of a fragmentation protocol that can be used to transmit an I/O Message greater than eight (8) bytes in length, DeviceNet does NOT define any protocol related information within the Data Field of an I/O Message.

Figure 2-8.1 Data Field of an I/O Message



Transmission of an I/O Message greater than eight (8) bytes in length is possible using the DeviceNet Fragmentation Protocol described in the next section.

## 2-9 Fragmentation/Reassembly

This section defines the means by which a message whose length is greater than 8 bytes (the maximum sized CAN frame) is fragmented and reassembled. The fragmentation/reassembly function is provided by the DeviceNet Connection Object and, as such, this section is actually a component of the DeviceNet Connection Object definition.

**Important:** Support for the transmission and reception of messages in a fragmented manner is optional.

The logic that triggers a fragmented transmission is different for Explicit Messaging Connections than it is for I/O Connections:

- *Explicit Messaging Connections* examine the **length of each message** to be transmitted. If the message is greater than eight bytes in length, then the fragmentation protocol is used.
- *I/O Connections* examine the **produced\_connection\_size attribute** of the Connection Object (see Volume 1, Chapter 3, section 3-4, Connection Object Class Definition). If the produced\_connection\_size attribute is greater than eight (8), then the fragmentation protocol is used.

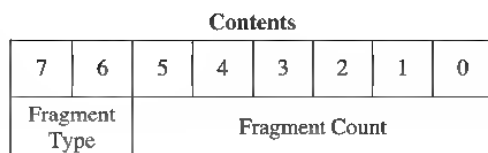
Two types of fragmentation are defined:

- **Acknowledged:** performed when fragmenting an Explicit Message
- **Unacknowledged:** performed when fragmenting an I/O Message

### 2-9.1 Fragmentation Protocol

The Fragmentation Protocol is located within a single byte in the CAN Data Field, and is formatted as follows:

Figure 2-9.1 Format of DeviceNet Fragmentation Protocol





### Fragmentation Protocol Contents:

**Fragment Type** - Indicates whether this is the first, middle, or last transmission. The following values are defined:

**Table 2-9.1 Fragment Type Bit Values**

Value	Meaning
0	First fragment. The Fragment Count Field must contain the value <b>0</b> or <b>3F</b> .
1	Middle fragment. <sup>2</sup>
2	Last fragment <sup>3</sup>
3	Fragment Acknowledge. <sup>4</sup>

- 1 If the Fragment Count contains the value zero (0), then this is the first in a series of fragments. If the Fragment Count field contains the value 3F, then this is also the last transmission in the series. This may occur with I/O Connections in which a large connection size was established, but only a small amount of data is currently being sent. The receiver needs to be told that this is both the first & last fragment.
- 2 Indicates that this is a “mid-stream” fragment. This fragment is neither the first nor the last fragment in the series. More fragments are forthcoming.
- 3 Marks this as the last fragment. This is used after having transmitted one or more previous fragments.
- 4 The value the receiver of a fragmented message uses to acknowledge the reception of a fragment.

**Fragment Count** - Marks each separate fragment such that the receiver can determine whether or not a fragment has been missed. If the Fragment Type is *First Fragment*, then this field takes on a special meaning (see Table 2-9.1 Fragment Type Bit Values). The Fragment Count is increased by one (1) for each successive fragment in a series and wraps back to zero (0) when it reaches the value 64 ( $\text{fragmentCount} = (\text{fragmentCount} + 1) \bmod 64$ ).

Placement of the Fragmentation Protocol is different within I/O Messages than it is within Explicit Messages. For **I/O Message** fragmentation, the Fragmentation Protocol information is placed within byte offset **zero (0)**. See Figure 2-9.2.

**Figure 2-9.2 I/O Message Fragment Format**

Byte Number	Contents							
	7	6	5	4	3	2	1	0
0	Fragment Type		Fragment Count					
	I/O Message Fragment							

For **Explicit Message** fragmentation, the Fragmentation Protocol information is placed within byte offset **one (1)**. See Figure 2-9.3.

**Figure 2-9.3 Explicit Message Fragment Format**

Byte Number	Contents							
	7	6	5	4	3	2	1	0
0	Frag [1]	XID	MAC ID					
1	Fragment Type		Fragment Count					
	Explicit Message Body Fragment							

Notice the **Frag** bit within the Message Header in Figure 2-9.3 is set to one (1) to indicate that this is only a piece of the Explicit Message, not the entire message. The value one (1) here also indicates that the next byte contains the Fragmentation Protocol.

The receiver of a fragmented series of transmissions parses the Fragmentation Protocol as defined in this section. This procedure applies to both I/O and Explicit Message fragmentation.

- **If the first transmission is expected** by the connection, then the Fragment Type must be equal to *First Fragment*. If the Fragment Count is 3F, then this is the **ONLY** transmission in the series, and the Connection does the following:
  - processes the message, and
  - resets to looking for the beginning of a new series.

If the Fragment Count is 0, then this is the **FIRST** in a series of transmissions and the Connection stores the fragment and saves the Fragment Count.

- **If the first transmission is NOT expected** by the Connection, then the Connection verifies that the Fragment Type is not *First Fragment*, and that the Fragment Count is numerically one (1) greater than the previous value received. If one of these checks fails, then an error has been detected. If both checks pass, then the fragment is appended to the previously received fragment(s), and the Fragment Type is parsed to determine whether or not more fragments are to be expected.

If more fragments are forthcoming, then the Connection saves the received Fragment Count and waits for the next fragment. If this is the last transmission in the series and an error has not been detected, then the Connection does the following:

- processes the message, and
- resets to looking for the beginning of a new series.

If an Explicit Message is being fragmented, then the receiver must generate and transmit an *acknowledgment* after the reception of each fragment.

If an error is detected, then error recovery specific to whether this is a fragmented I/O or Explicit Message takes place.

If the detection of a missed fragment was triggered by the reception of the first fragment in the next series (Fragment Type = *First Fragment*), then any processing associated with the current series is immediately discontinued, the fragments stored in memory are discarded, and processing immediately begins on the new series. If this scenario occurs when fragmenting an Explicit Message, then the receiver acknowledges the first transmission in the next series and discards the previous series as noted.

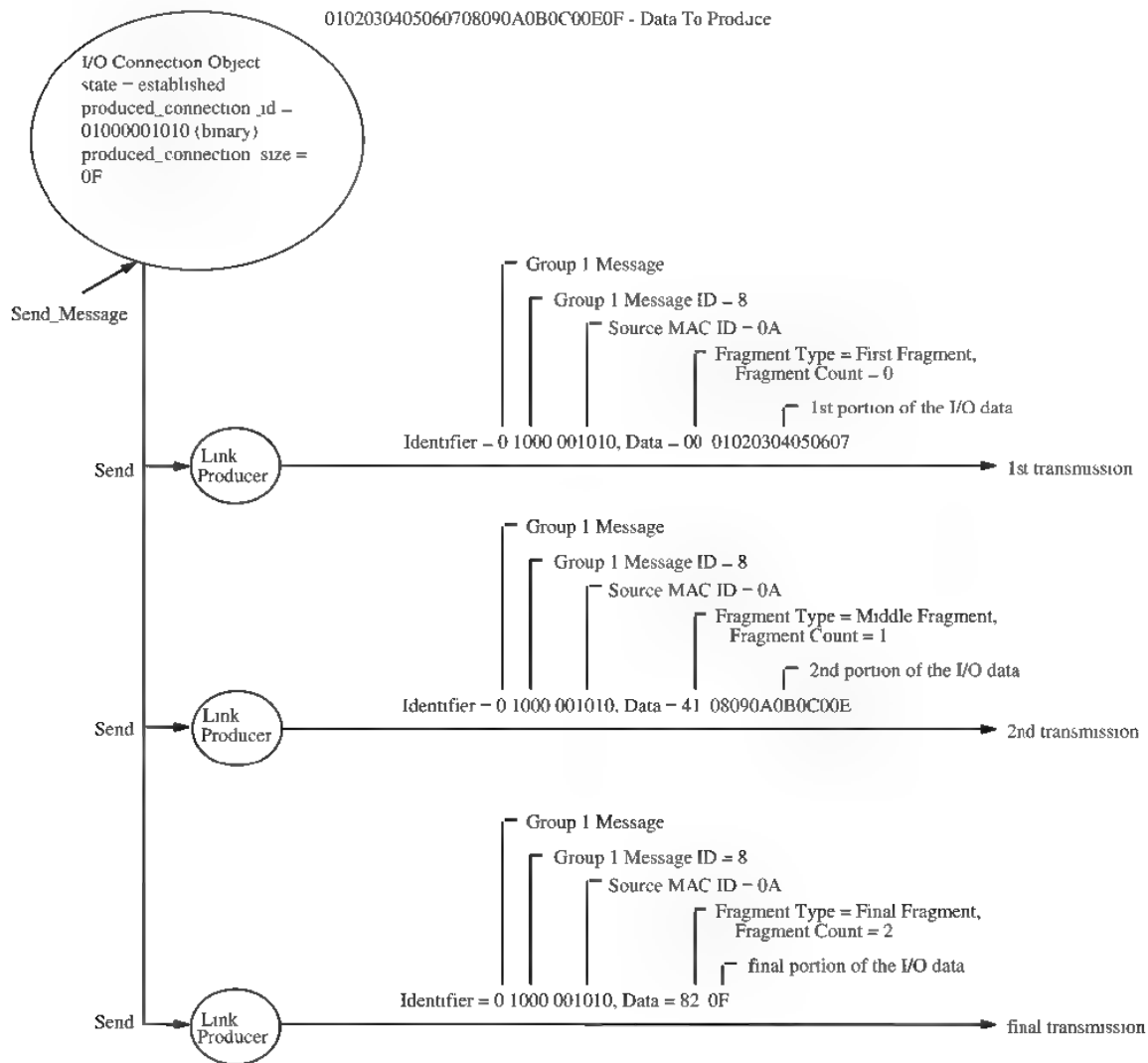
## **2-9.2 Unacknowledged Fragmentation**

Fragmentation of an **I/O Message** is performed in an *Unacknowledged* fashion. Unacknowledged fragmentation consists of the back-to-back transmission of the fragments from the transmitting module. The receiving module(s) returns no acknowledgments (other than the CAN-provided Ack) on a per-fragment basis. The Connection simply invokes the Link Producer's Send service as necessary to move the message without waiting for any specific acknowledgment from the receiving module(s).

When an I/O Connection's Send\_Message service is invoked, it examines its **connection\_size** attribute to determine whether or not a fragmented series of messages is to be transmitted. As previously stated, if the **connection\_size** attribute is greater than eight (8), then the Fragmentation Protocol is placed within the I/O Message, regardless of the length of the piece of I/O currently being transmitted. Figure 2-9.4 depicts the fragmented transmission of a 15-byte I/O Message.

**Figure 2-9.4 Fragmented Transmission of a 15-Byte I/O Message**

Producing module's MAC ID – 0A Assume that the I/O Connection has been established across Group 1



If the connection\_size attribute is less than or equal to eight (8) bytes, then the raw data is transmitted without the presence of the Fragmentation Protocol. See Figure 2-9.5.

**Figure 2-9.5 Transmission of raw data without the Fragmentation Protocol**

Producing module's MAC ID = 0A. Assume that the I/O Connection has been established across Group 1.

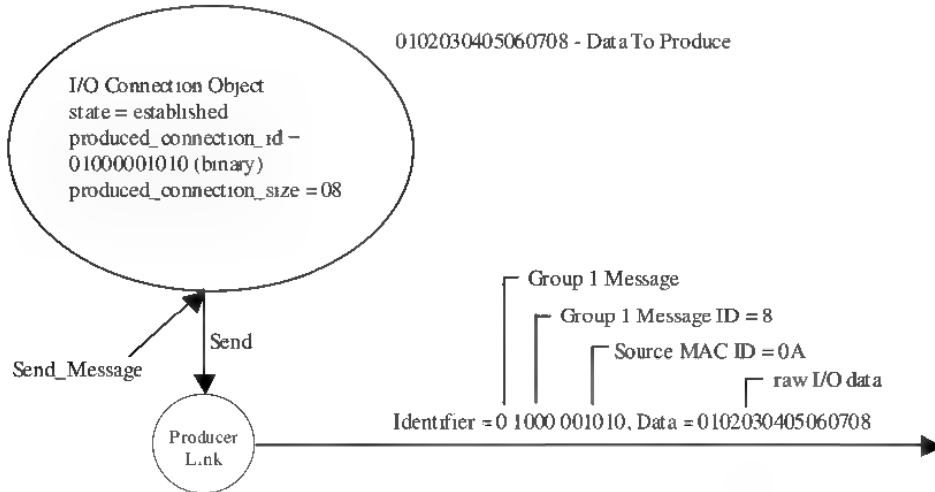
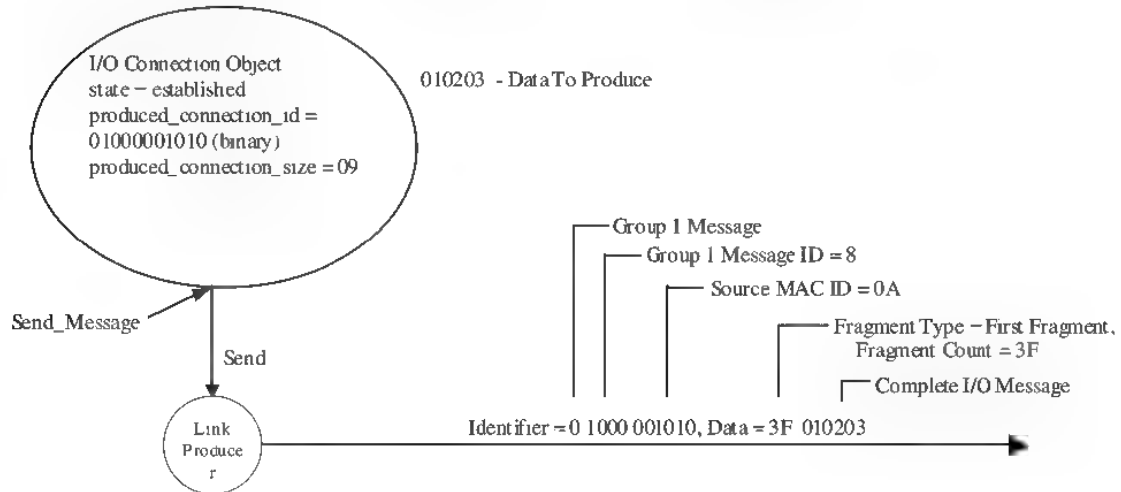


Figure 2-9.6 illustrates that the Fragmentation Protocol within the I/O Message is specified based on the **connection\_size** attribute, not based on the length of the current piece of I/O data to transmit.

**Figure 2-9.6 Specification of the Fragmentation Protocol based on the connection\_size attribute**

Producing module's MAC ID = 0A. Assume that the I/O Connection has been established across Group 1. Assume that when the I/O Connection was established the connection\_size attribute was initialized to 9 but currently only 3 bytes of data are to be transmitted.

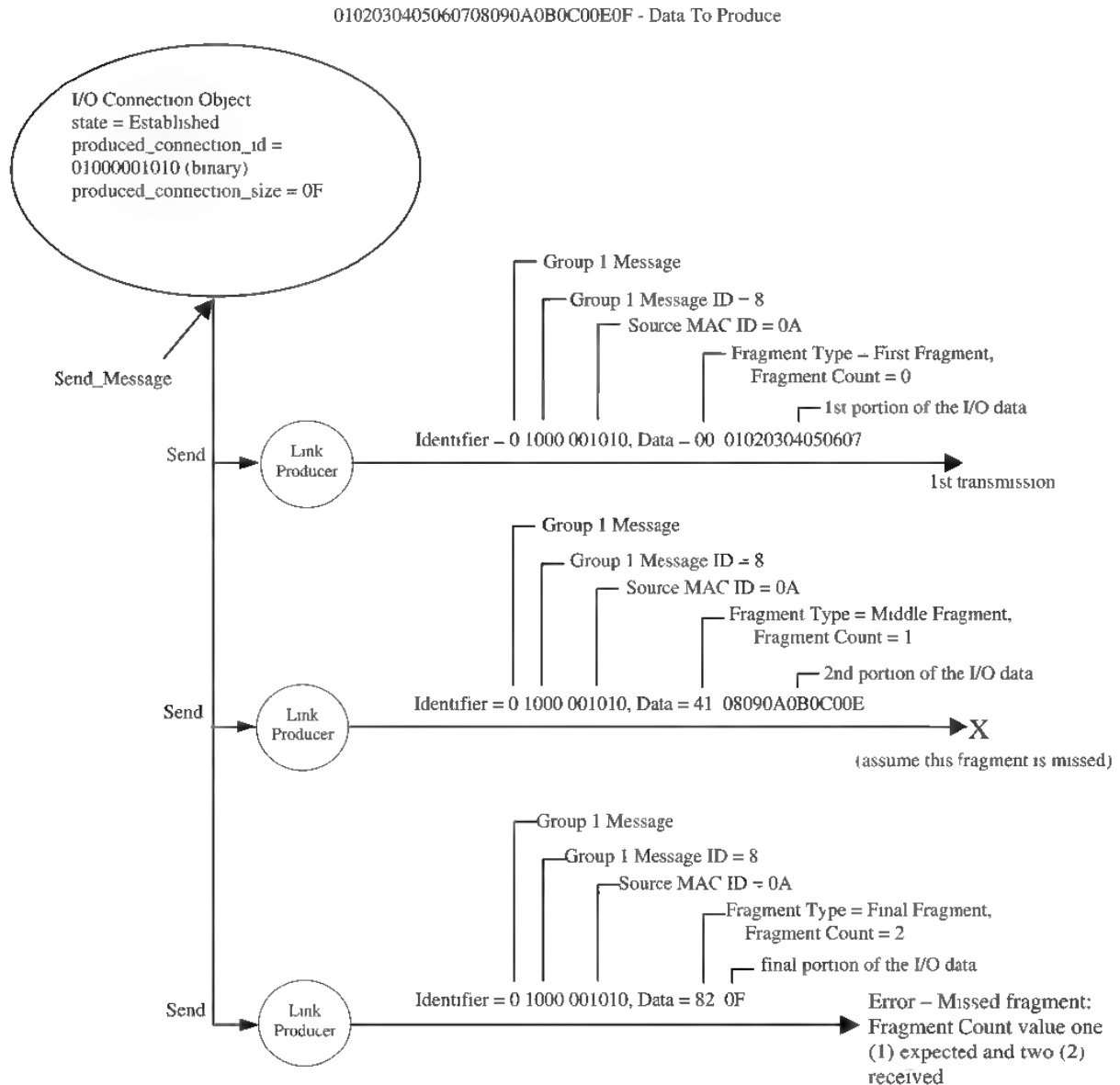


If the Application requests to transmit a piece of data whose length is greater than the **connection\_size** attribute, then an internal error is indicated and the transmission does not occur. If the receiving I/O Connection Object detects a missed fragment by determining that the received Fragment Count is not equal to the previously received Fragment Count plus one (1), then the following *error recovery* steps are taken:

- All subsequent fragments **in this series** are dropped and the Application is not informed of an I/O Message reception.
- The Connection Object begins looking for the beginning of a new fragmented series of transmissions while throwing away the remaining fragments in this series.

**Figure 2-9.7 I/O Fragment Error Handling**

Producing module's MAC ID = 0A. Assume that the I/O Connection has been established across Group 1.



The Event/Action Matrices presented below provide formal definitions of the transmitting and receiving sides of the Unacknowledged Fragmentation Protocol.

**Table 2-9.2 Unacknowledged Fragmented Transmission**

Event	Condition	Action
Sending First Fragment	This is also the last fragment	Fragment Count = 0x3f Fragment Type = First Fragment Build and transmit the message.
	Subsequent fragments will be transmitted.	Fragment Count = 0. Fragment Type = First Fragment. Build and transmit the message fragment. Continue building/transmitting per the Event column
Sending "mid-stream" fragment (not the first & not the last).	None	Increment previously transmitted Fragment Count by one (1). Fragment Type = Middle Fragment Build and transmit the message fragment. Continue building/transmitting per the Event column.
Sending final fragment	One or more previous fragments have already been transmitted	Increment Fragment Count by one (1). Fragment Type = Final Fragment. Build and transmit the message fragment. Continue building/transmitting per the Event column.

**Table 2-9.3 Unacknowledged Fragmented Reception**

Event	Condition	Action
Received Fragment, Fragment Type = First Fragment	Fragment Count = 0	If processing was in progress with a previous fragmented message, discontinue that processing. Store the Fragment Count and associated message fragment
	Fragment Count = 0x3f	If processing was in progress with a previous fragmented message, discontinue that processing. Deliver the associated message data to the Application Object and reset to waiting for the First Fragment of the next transmission.
	Fragment Count <b>IS NOT</b> equal to 0x00 or 0x3f	If processing was in progress with a previous fragmented message, discontinue that processing. Discard the fragment and reset to waiting for the First Fragment of the next transmission
Received Fragment, Fragment Type = Middle Fragment	The First Fragment <b>HAS NOT</b> yet been received	Discard the message fragment. Continue to wait for the first fragment.
	The Fragment Count is numerically one (1) greater than the previously received Fragment Count.	Store the Fragment Count and associated message fragment.
	The Fragment Count <b>IS NOT</b> numerically one (1) greater than the previously received Fragment Count	Discard the fragment and reset to waiting for the First Fragment of the next transmission.
	Too much data has been received. More bytes than the <b>consumed_connection_size</b> attribute allows for have been received.	Discard the fragment and reset to waiting for the First Fragment of the next transmission

Event	Condition	Action
Received Fragment, Fragment Type = Final Fragment	The First Fragment <b>HAS NOT</b> yet been received.	Discard the message fragment. Continue to wait for the first fragment.
	The Fragment Count is numerically one (1) greater than the previously received Fragment Count	Deliver the reassembled message data to the Application Object and reset to waiting for the First Fragment of the next transmission.
	The Fragment Count <b>IS NOT</b> numerically one (1) greater than the previously received Fragment Count	Discard the fragment and reset to waiting for the First Fragment of the next transmission
	Too much data has been received. More bytes than the <b>consumed_connection_size</b> attribute allows for have been received.	Discard the fragment and reset to waiting for the First Fragment of the next transmission.
Received Fragment, Fragment Type = Fragment Acknowledge	None	Discard the fragment and reset to waiting for the First Fragment of the next transmission.

### 2-9.3 Acknowledged Fragmentation

Fragmentation of an **Explicit Message** is performed in an **Acknowledged** fashion. Acknowledged fragmentation consists of the transmission of a fragment from the transmitting module followed by the transmission of an acknowledgment by the receiving module. The receiving module acknowledges the reception of **each** fragment. This provides a degree of flow control. The assumption is that larger bodies of information may be moved across Explicit Messaging Connections (e.g. Upload/Download functions) and, as such, a degree of flow control is necessary.

Figure 2-9.8 illustrates the format for the acknowledgment/response message transmitted by the receiver after **each** Explicit Message fragment is received.

**Figure 2-9.8 Acknowledgment/Response Message Format**

Byte Number	Contents							
	7	6	5	4	3	2	1	0
0	Frag [1]	XID	MAC ID					
1	Fragment Type (3)		Fragment Count					
2	Ack Status							



### 2-9.3.1 Acknowledgment/Response Data Frame Contents

**Fragment Type** - Indicates that this is a *Fragment Acknowledge* by placing the value 3 within this field.

**Fragment Count** - Echoes the last Fragment Count value received.

**Ack Status** - Indicates whether or not an error has been encountered by the receiver of the fragmented message. The following values are defined:

**Table 2-9.4 Ack Status Bit Values**

Value	Meaning
0	Success. No errors have been detected & the fragmented transmission should continue.
1	Too Much Data. The maximum amount of data the receiver can receive across this Connection has been exceeded
2 - FF	Reserved by DeviceNet for future use

Notice the **Frag** bit within the Message Header in Figure 2-9.8 is set to one (1) in the acknowledgment frame, even though the acknowledgment is only three bytes in length. Setting the Frag bit to one (1) is necessary to trigger the Connection Object to interpret the next byte as the Fragmentation Protocol instead of performing normal Explicit Message handling.

The **transmitting** side functions as specified below:

- The Connection Object formulates the Message Header by setting the Frag bit to one (1), the XID (Transaction ID) field to the value specified by the Application, and initializes the MAC ID field in the same manner it would a non-fragmented Explicit Message. The Message Headers associated with each separate fragment are identical.
- The Connection Object then places the appropriate Fragmentation Protocol information into the message. The Connection Object stores the Fragment Count that was inserted into the message.
- The Connection Object then takes the next piece of the Message Body and places it into the message.
- The message is transmitted and a *wait for acknowledgment* timer is started. The amount of time to wait is Application-specific.
- If the *wait for acknowledgment* timer expires, then the Connection Object automatically retries the last transmission. **The Connection Object retries once.** If the timer expires once again (two consecutive *wait for acknowledgment* timeouts), then the Application is informed that an error has been detected and the requested transmission cannot take place.
- If an acknowledgment is received, then the Connection Object determines whether or not the Fragment Count in the acknowledgment is equal to the last Fragment Count it transmitted. If they are equal, then the fragment was successfully delivered and acknowledged, and normal processing continues. If the values are not equal, then the Connection Object continues to wait for an acknowledgment with a matching Fragment Count.

The *initial state* associated with the receiving side entails waiting for either the First Fragment in a fragmented transmission or waiting for the reception of a complete Explicit Message. The **receiving** side functions as specified below:

- If the Message Header indicates that this is a fragmented portion of an Explicit Message, then the Connection Object examines the Fragmentation Protocol to determine its validity. If the Connection has yet to receive the first transmission in the series (in the *initial state*) and the Fragment Type field is not equal to *First Fragment*, then the fragment is dropped and no acknowledgment is returned.

- If the Fragment Type indicates that this is the *First Fragment*, then the Fragment Count must equal zero (0). If this is the case, then the message fragment is stored and an acknowledgment is returned. If the Fragment Type indicates *First Fragment* and the Fragment Count is not zero (0), then the fragment is dropped and no acknowledgment is returned.
- If the Fragment Count is one (1) greater than the previously received count and the Fragment Type does not indicate that this is the First Fragment, then the next fragment has been received. The fragment is appended to the previously received fragment(s) and an acknowledgment is returned. The Fragment Count associated with this fragment is stored.
- If the Fragment Count is neither one (1) greater nor equal to the previously received count, then the fragment is discarded and no acknowledgment is returned. The receiver resets to the *initial state*.
- When the final fragment is received and the acknowledgment is transmitted, the Connection Object continues processing the message as if it were non-fragmented.

The following example illustrates a fragmented response from a Server to the Client. The Connection, Link Producer, and Link Consumer Objects are not illustrated.

**Figure 2-9.9 Fragmented Response from a Server to the Client**

Client's MAC ID = 0A, Servers MAC ID = 02. Assume that the Explicit Messaging Connection has been established across Group 1 and the Message Body Format established is DeviceNet (8/8). Assume the attribute data = 0102030405060708090A0B0C0D0E0F

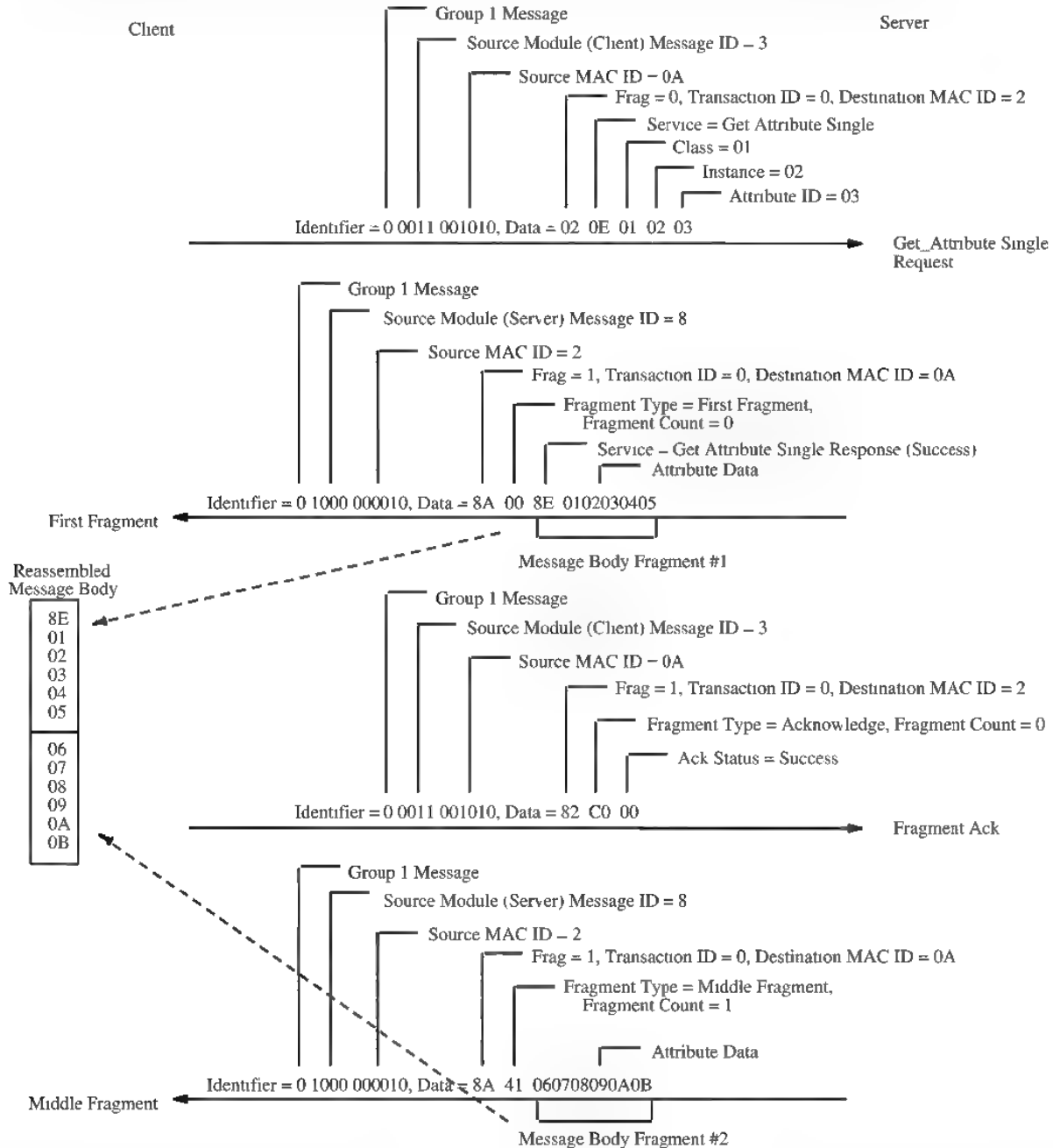




Figure 2-9.11 Fragmented Request from the Client to the Server

Client's MAC ID = 0A, Servers MAC ID = 02. Assume that the Explicit Messaging Connection has been established across Group 1 and the Message Body Format established is DeviceNet (8/8). Assume the attribute data – 0102030405060708090A0B0C0D0E0F

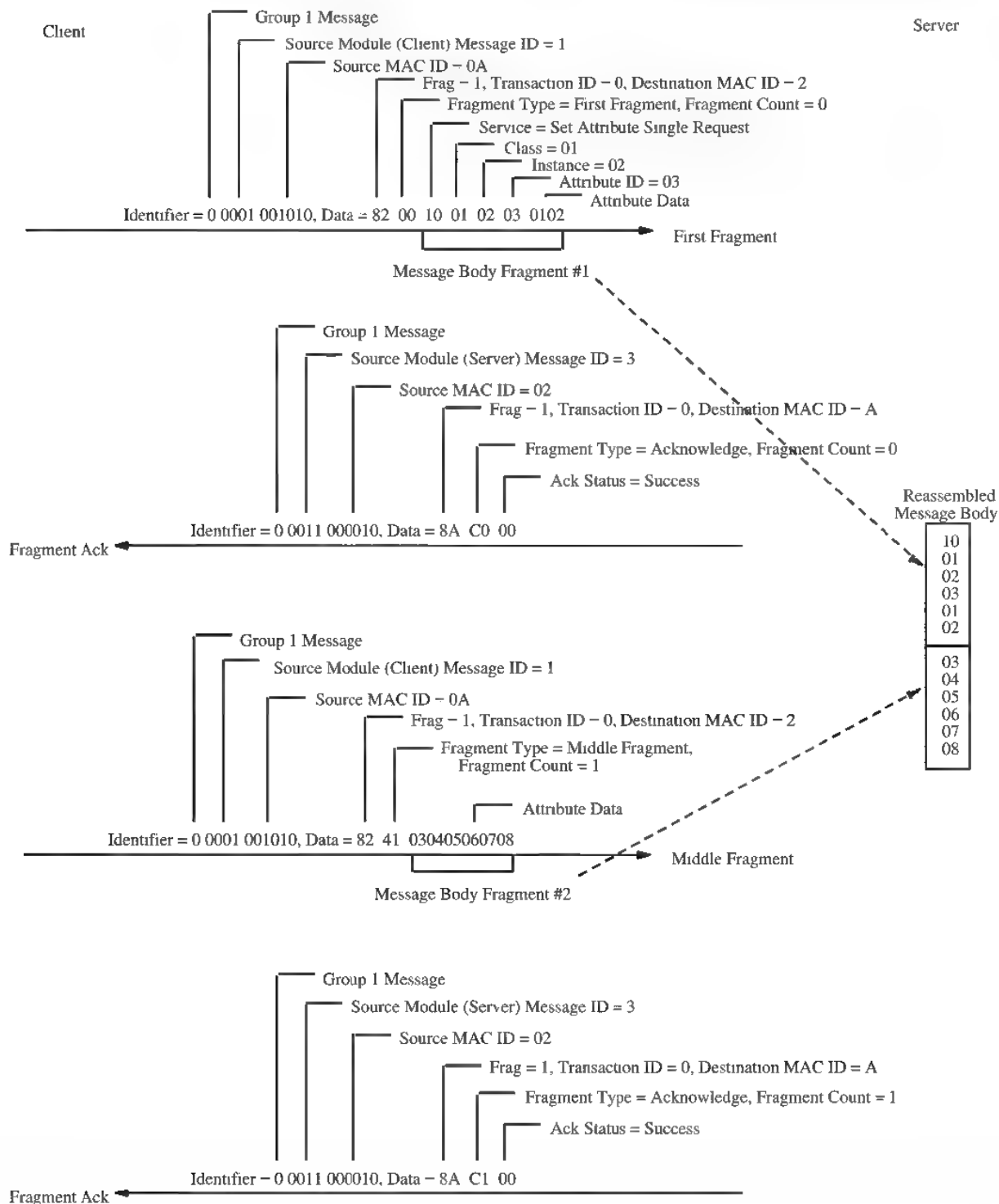
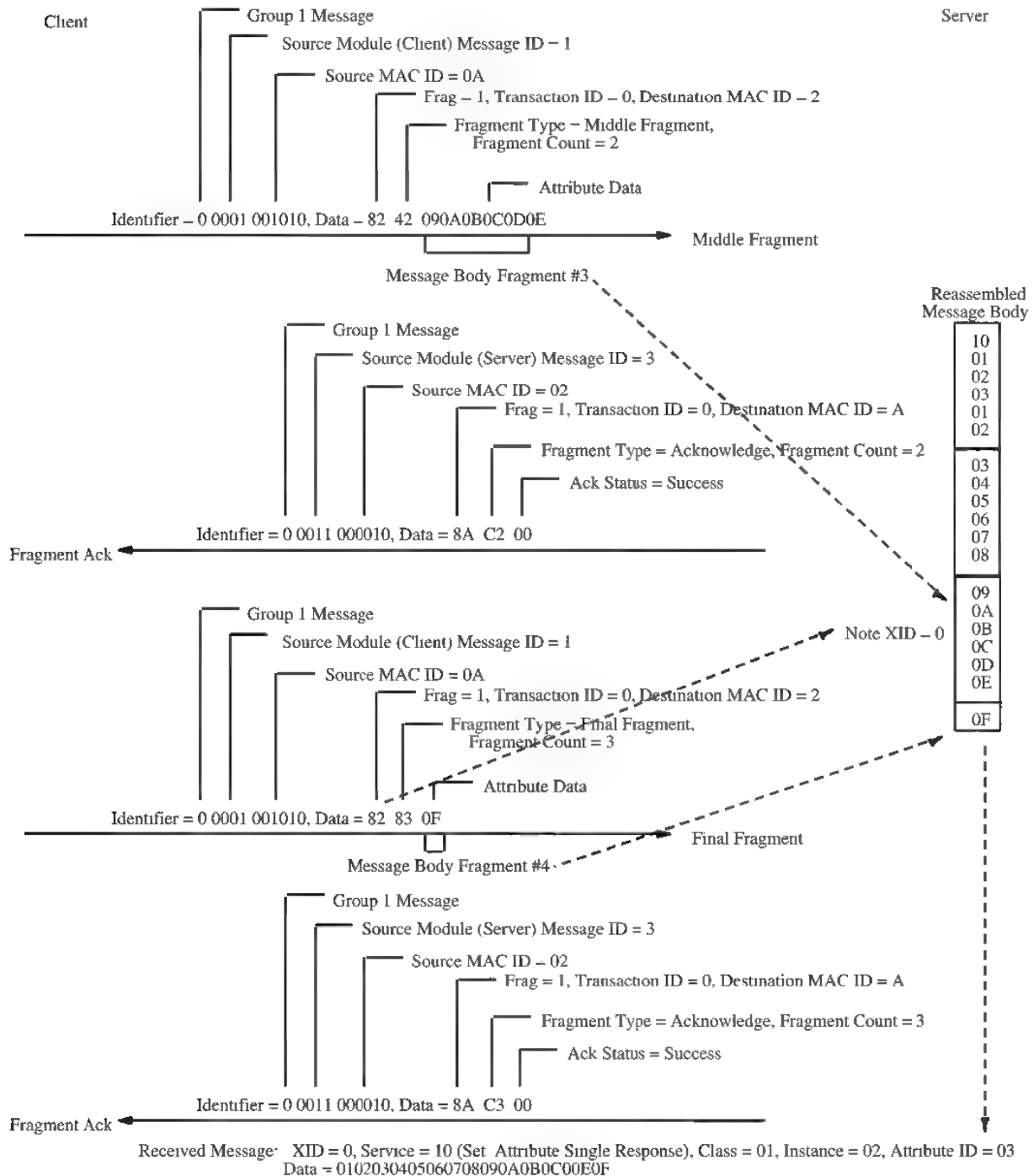


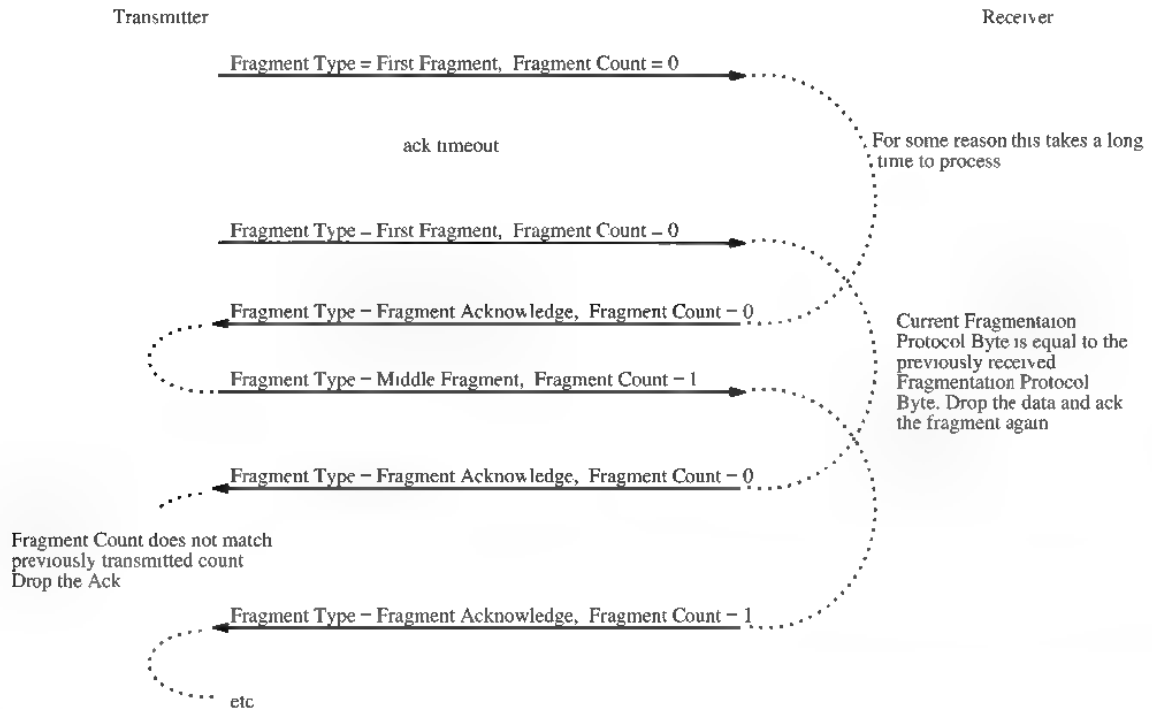
Figure 2-9.12 Fragmented Request from the Client to the Server (cont'd)



**Important:** Note that following the reception of the Set\_Attribute Single Request in a fragmented manner, the Server processes the complete message and returns the associated response just as if the Set\_Attribute Single Request had been received in a non-fragmented manner.

The illustration below depicts an Acknowledgment timeout and recovery during a fragmented transfer.

Figure 2-9.13 Timeouts when Performing a Fragmented Explicit Message Transfer



The Event/Action Matrices presented below provide formal definitions of the transmitting and receiving sides of the Acknowledged Fragmentation Protocol.

Table 2-9.5 Acknowledged Fragmented Transmission

Event	Condition	Action
Sending First Fragment	None	Fragment Count = 0. Fragment Type = First Fragment. Build and transmit the message. Wait for the Fragment Acknowledgment to be returned.
Time- out while waiting for a Fragment Acknowledge occurs	This is the first time-out associated with this particular Acknowledgment	Re-transmit the previously sent message fragment. Wait for the Fragment Acknowledge to be returned
	Second <b>consecutive</b> time- out while waiting for the Fragment Acknowledge occurs	Discontinue the attempt to transmit this message. Issue the necessary internal indications.
Receive Fragment Acknowledge associated with either the First Fragment or a Middle Fragment	Fragment Count in the Ack <b>DOES NOT</b> match the previously transmitted Fragment Count	Discard the received Acknowledgment. Continue waiting for the appropriate Acknowledgment.
	Fragment Count in the Ack matches the previously transmitted Fragment Count but the Ack Status <b>IS NOT</b> equal to zero (0).	Discontinue the attempt to transmit this message. Issue the necessary internal indications
	Fragment Count in the Ack matches the previously transmitted Fragment Count and the Ack Status <b>IS</b> equal to zero (0).	Build and transmit the next fragment per the Event column

Event	Condition	Action
Receive Fragment Acknowledge associated with the Final Fragment	Fragment Count in the Ack <b>DOES NOT</b> match the previously transmitted Fragment Count	Discard the received Acknowledgment. Continue waiting for the appropriate Acknowledgment.
	Fragment Count in the Ack matches the previously transmitted Fragment Count but the Ack Status <b>IS NOT</b> equal to zero (0).	Discontinue the attempt to transmit this message. Issue the necessary internal indications.
	Fragment Count in the Ack matches the previously transmitted Fragment Count and the Ack Status <b>IS</b> equal to zero (0).	Indicate that the message has been successfully delivered to the remote destination.
Sending "mid-stream" fragment (not the first & not the last).	None	Increment previously transmitted Fragment Count by one (1). Fragment Type = Middle Fragment. Build and transmit the message fragment. Wait for the Fragment Acknowledgment to be returned.
Sending final fragment	None	Increment Fragment Count by one (1). Fragment Type = Final Fragment. Build and transmit the message fragment. Wait for the Fragment Acknowledgment to be returned.
The final fragment of an Explicit Request Message has been sent.	The associated response message is received prior to receiving the Acknowledgment associated with the Final Fragment	See section 2-10.2.

Note that the *initial state* referenced Table 2-9.6 below is defined as waiting for either the First Fragmented in a fragmented transmission or waiting for the reception of a complete Explicit Message (non-fragmented).

**Table 2-9.6 Acknowledged Fragmented Reception**

Event	Condition	Action
Receive First Fragment	Receiving side of Acknowledged Fragmentation is <b>NOT</b> supported.	Return Fragment Acknowledge with Ack Status set to one (1). Remain in the <i>initial state</i> .
	Fragment Count <b>IS NOT</b> equal to zero (0).	Discard the fragment and reset to the <i>initial state</i> .
	Fragment Count = 0.	Store the Fragmentation Protocol Byte and associated message fragment. If the Connection was in the process of receiving a previous fragmented message that had yet to be completed, then that message is discarded and the Connection begins processing this new fragmented message. Return the Fragment Acknowledge with the Ack Status set to <i>success</i> .



Event	Condition	Action
Received Fragment Type = Middle Fragment.	The First Fragment <b>HAS NOT</b> yet been received.	Discard the message fragment.
	The Fragment Count is numerically one (1) greater than the previously received Fragment Count	Store the Fragment Count and associated message fragment. Return the Fragment Acknowledge with the Ack Status set to <i>success</i>
	The Fragmentation Protocol Byte (both the Fragment Type and Fragment Count) is equal to the previously received Fragmentation Protocol Byte.	Re-Acknowledge the fragment by returning the previously transmitted Acknowledgment.
	The Fragment Count is neither one (1) greater than nor equal to the previously received Fragment Count.	Discard the fragment and reset to the <i>initial state</i>
	Too much data has been received.	Return the Fragment Acknowledge with the Ack Status set to indicate the error that has occurred.
Received Fragment Type = Final Fragment.	The First Fragment <b>HAS NOT</b> yet been received.	Discard the message fragment
	The Fragment Count is numerically one (1) greater than the previously received Fragment Count	Return the Fragment Acknowledge with the Ack Status set to <i>success</i> . Process the received Explicit Message
	The Fragmentation Protocol Byte (both the Fragment Type and Fragment Count) is equal to the previously received Fragmentation Protocol Byte	Discard the message fragment Re-Acknowledge the fragment by returning the previously transmitted Acknowledgment.
	The Fragment Count is neither one (1) greater than nor equal to the previously received Fragment Count.	Discard the fragment and reset to the <i>initial state</i> .
	Too much data has been received	Return the Fragment Acknowledge with the Ack Status set to indicate the error that has occurred.
Receive non-fragmented message	In the process of <b>receiving</b> a fragmented message	Discontinue processing associated with the fragmented message. Process the received non-fragmented message. Reset to the <i>initial state</i>

See section 2-10.2 for additional considerations relative to Acknowledged Fragmentation.

## 2-10 Explicit Messaging Client Application considerations

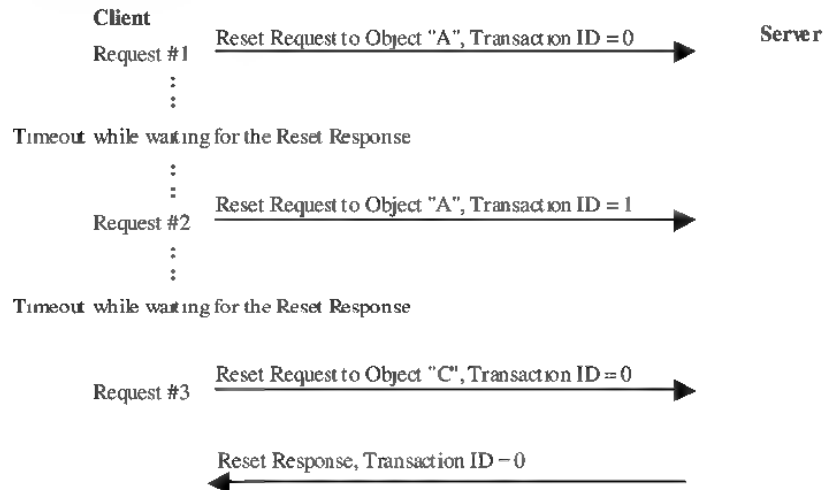
This section lists considerations that may effect an implementation that includes the ability to send Explicit Request Messages (behave as a Client) across an Explicit Messaging Connection.

### 2-10.1 Request/Response Timeouts

The Client Application is responsible for providing response timeout facilities. The amount of time a Client waits for a Server to respond to a request depends on the application and, possibly, on the service. Due to the Media Access mechanism employed by CAN, an implementation should wait until an Explicit Request Message is actually transmitted on the network before activating any response related timers.

Two consecutive timeout events while waiting for an Explicit Response message brings about the following scenario:

**Figure 2-10.1 Request/Response Timeout Handling**



In this case the Client Application cannot be certain which Reset Request the Reset Response is associated with. It could be associated with either Request #1 or Request #3. This scenario is a result of having a single bit in which the Transaction ID field resides.

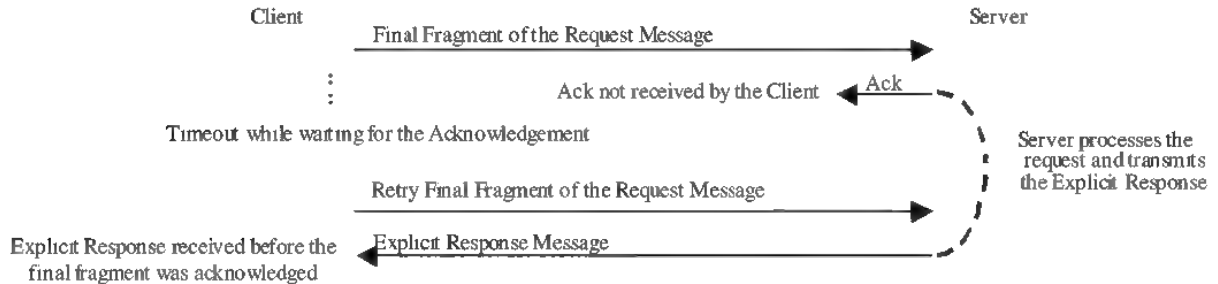
The Client Application must ensure that it is synchronized with the Server following this timeout scenario. In some cases the Inactivity/Watchdog Timer will automatically handle this scenario by expiring and causing the deletion of the Connection Object(s).

**Important:** The safest approach to handling this scenario is to close the current Messaging Connection and establish/use a new Explicit Messaging Connection.

## 2-10.2 Explicit Request Message Fragmentation Timeouts

The following scenario related to the transfer of the Final Fragment associated with an Explicit Request Message exists:

**Figure 2-10.2 Final Fragment Ack Timeout**

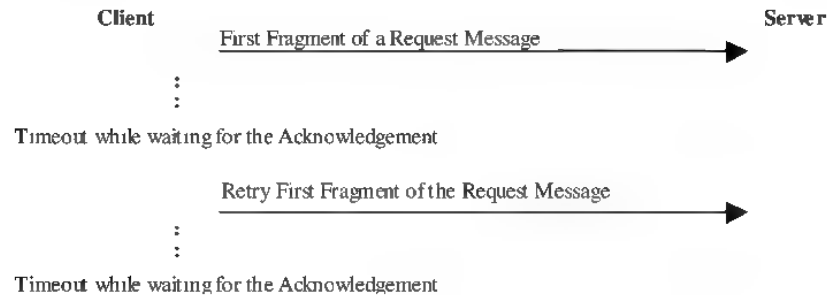


An Explicit Messaging Client Application has two choices relative to handling the scenario illustrated in Figure 2-10.2.

- It can strictly look for an Acknowledgment to the last fragment and, if it is not received, it can assume the full Explicit Request WAS NOT properly transferred.
- It can utilize the Explicit Messaging Response as an indication that the request was actually received. This indicates that the Acknowledgement of the final fragment was not received by the Client. In this case the Explicit Messaging Response is also serving as an Acknowledgement of the final fragment.

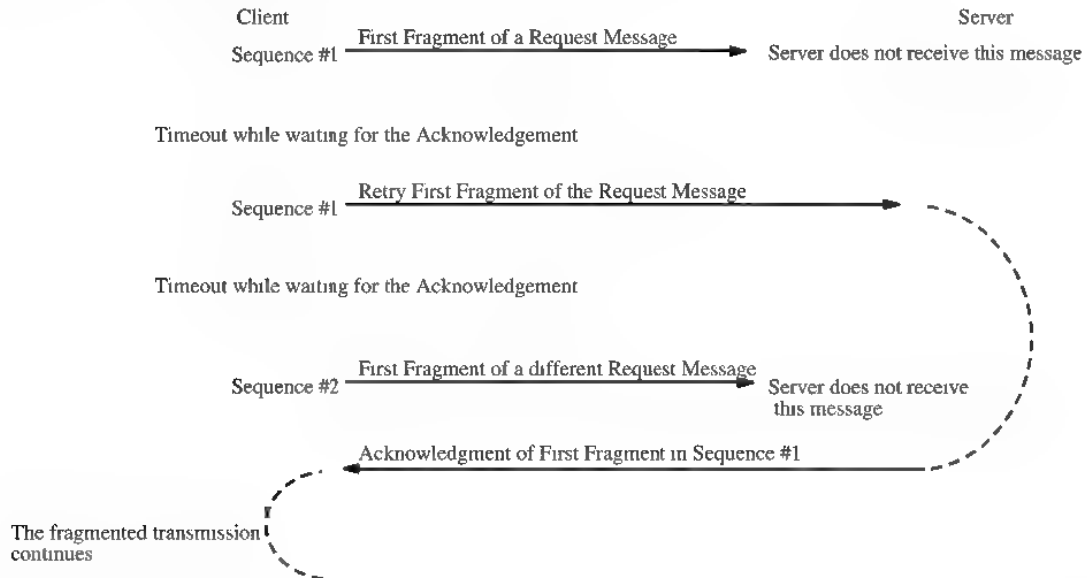
Figure 2-10.3 illustrates two consecutive timeouts while waiting for the Acknowledgement associated with the First Fragment of an Explicit Request Message:

**Figure 2-10.3 First Fragment Ack Timeouts**



Under these conditions the Client Application CANNOT immediately initiate the fragmented transfer of a different Explicit Request Message across the same Explicit Messaging Connection. This is due to the possibility of the scenario presented in Figure 2-10.4 below:

**Figure 2-10.4 First Fragment Ack Timeouts**



In this case, the Acknowledgment is erroneously interpreted by the Client as an Ack of the First Fragment in Sequence #2. The Client Application has the following options if this timeout scenario occurs:

- Try to send the same Explicit Request once again.
- Attempt to transmit a non-fragmented Explicit Request
- Close the current connection and establish/use a new Explicit Messaging Connection.

Note that the Overload condition described in CAN documentation is one possible cause of the *Message Not Received* event illustrated in the previous series of figures.

The Client Application must ensure that it is synchronized with the Server following this timeout scenario. In some cases the Inactivity/Watchdog Timer will automatically handle the scenarios presented in this section by expiring and causing the deletion of the Connection Object(s).

## 2-11 Offline Connection Set

This section describes the *Offline Connection Set* Messaging Protocol and presents details associated with the establishment of *Offline Connection Set* ownership. Support of the *Offline Connection Set* is optional for all types of devices. The Group 4 *Offline Connection Set* messages are used by client tool(s) to recover nodes in the *Communication Faulted* state. Using the *Offline Connection Set* messages a client (tool) shall be able to:

1. visually identify the faulted node(s) to which it is communicating with by flashing an LED,
2. send fault recovery messages to the faulted node, and when possible
3. recover the faulted node without having to unplug it from the subnet.

At any point in time, only one (1) client node may communicate with nodes in the *Communication Faulted* state, connected to a single subnet. Ownership of *Communication Faulted* nodes is gained via a dialog between clients (tools) using the *Offline Ownership Request/Response Messages*. See section 2-11.1 for a description of the *Offline Ownership* process.

The following table shows the Group 4 Identifiers associated with the *Offline Connection Set*.

**Figure 2-11.1 Offline Connection Set**

IDENTIFIER BITS (ConnectionID)											IDENTITY USAGE	
Group ID					Message ID							
10	9	8	7	6	5	4	3	2	1	0	Group 4 Messages	
1	1	1	1	1	2C							Communication Faulted Response Message
1	1	1	1	1	2D							Communication Faulted Request Message
1	1	1	1	1	2E							Offline Ownership Response Message
1	1	1	1	1	2F							Offline Ownership Request Message

Only clients wishing to support the *Off-line Connection Set* shall produce messages using *Group 4 MessageID* 2F and consume response messages with *Group 4 MessageID* 2E. Once ownership is gained, the client shall produce all messages destined for *Communication Faulted* node(s) using *Group 4 MessageID* 2D.

**NOTE:** A client may NOT produce a “*Communication Faulted Request Message*” until it has gained ownership of the “*Off-line Connection Set*”.

Once a client (tool) establishes *Offline Connection Set* ownership, it is able to transmit *Communication Faulted Request* messages with *Group 4 Message ID* 2D and receive *Communication Faulted Response* messages with *Group 4 Message ID* 2C.

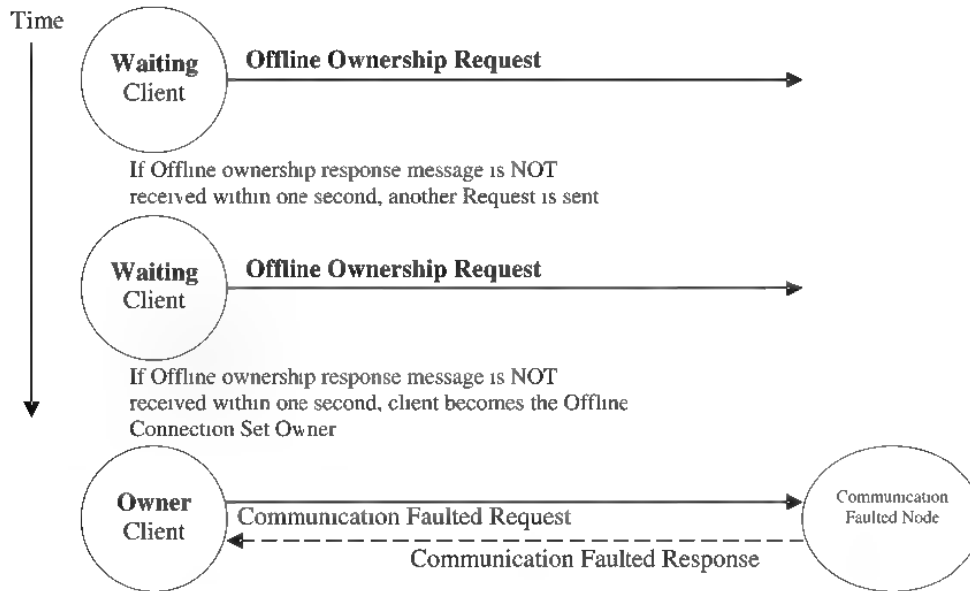
While in the *Communication Faulted* state, a node supporting this feature is only required to consume a single *ConnectionID*; *Group 4 MessageID* 2D. A faulted node shall produce its communication fault response messages on *Group 4 MessageID* 2C.

*Offline Connection Set* messages are low priority and may be subject to delays due to other network traffic.

### 2-11.1 Offline Ownership

Figure 2-11.2 illustrates the steps involved for a client (tool) to gain ownership of the Offline Connection Set.

Figure 2-11.2 Establishing the Offline Ownership

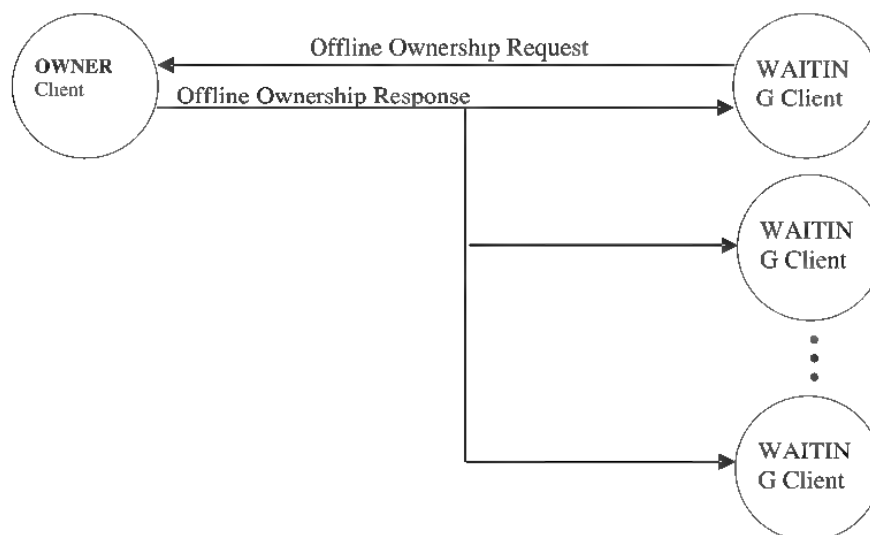


To gain control of the *Off-line Connection Set* a client (tool) shall produce its *Off-line Ownership Request Message*. Upon successful transmission, the client (tool) shall wait for an *Off-line Ownership Response Message* for a period of one second. If a response is not received, it shall produce a second *Off-line Ownership Request Message* and shall wait another second. If a response is not received, it shall become the owner of the *Off-line Request Message*. If an *Off-line Ownership Response Message* is received during either wait period, it shall not become the owner of the *Off-line Connection Set* and shall wait to become the owner.

**Note:** Only one client tool shall have ownership of the offline connection set at any point in time.

The scenario where one client (tool) has previously claimed ownership of the *Off-line Connection Set*, and additional client nodes are arbitrating for ownership, the current owner shall reply to any *Off-line Ownership Request Messages* with an *Off-line Ownership Response Message*. The respective dialogs are shown in Figure 2-11.3.

Figure 2-11.3 Multicast Nature of the Offline Ownership



A waiting Client shall not attempt to send an *Offline Ownership Request Message* for a minimum of two seconds after receiving a *Offline Ownership Response Message*.

## 2-11.2 Offline Ownership Messages

The two messages used to manage the Offline Connection Set are only produced and consumed by client nodes supporting this functionality. These messages are only serviced while a client is participating in a recovery activity, otherwise they are ignored by a client.

### 2-11.2.1 Offline Ownership Request Message (Client Only)

To gain ownership of the *Offline Connection Set*, a client tool shall produce an *Offline Ownership Request Message* with Group 4 Message ID 2F. The message protocol follows;

Figure 2-11.4 Offline Ownership Request Message

	Contents							
Byte Offset	7	6	5	4	3	2	1	0
0	Reserved [0]		Client MAC ID					
1	R/R [0]	Allocate [0x4B]						
2	Vendor ID						Low byte	
3							High byte	
4	Serial Number						Low byte	
5								
6								
7							High byte	

Upon successful transmission, the client (tool) shall wait for an *Offline Ownership Response* Message for at least one (1) second. If a response is not received, it shall produce a second *Offline Ownership Request* Message and shall wait at least one (1) second. If a response is not received, it shall become the owner of the *Offline Connection Set*. If an *Offline Ownership Response Message* is received during either time-period, it shall not become the owner of the *Offline Connection Set* and shall wait to become the owner.

Once a client is the owner of the *Offline Connection Set*, if it receives an *Offline Ownership Request* Message, it shall produce an *Offline Ownership Response* Message within one (1) second.

If a client is waiting to become the owner of the *Offline Connection Set*, it shall not produce an *Offline Ownership Request Message* at a rate faster than once every two seconds. This two-second delay shall be reset upon the receipt of any message using an *Offline Ownership Request* or *Response Connection\_ID*.

### 2-11.2.2 Offline Ownership Response Message (Client Only)

The format of the *Offline Ownership Response* Message is the same as the *Offline Ownership Request* Message, except it is produced at Group 4 Message ID 2E, and the R/R bit is set (1).

Figure 2-11.5 Offline Ownership Response Message Protocol

		Contents							
Byte Offset		7	6	5	4	3	2	1	0
0		Reserved [0]		Client MAC ID					
1		R/R [1]	Allocate [0x4B]						
2		Vendor ID							Low byte
3									High byte
4		Serial Number							Low byte
5									
6									
7									High byte

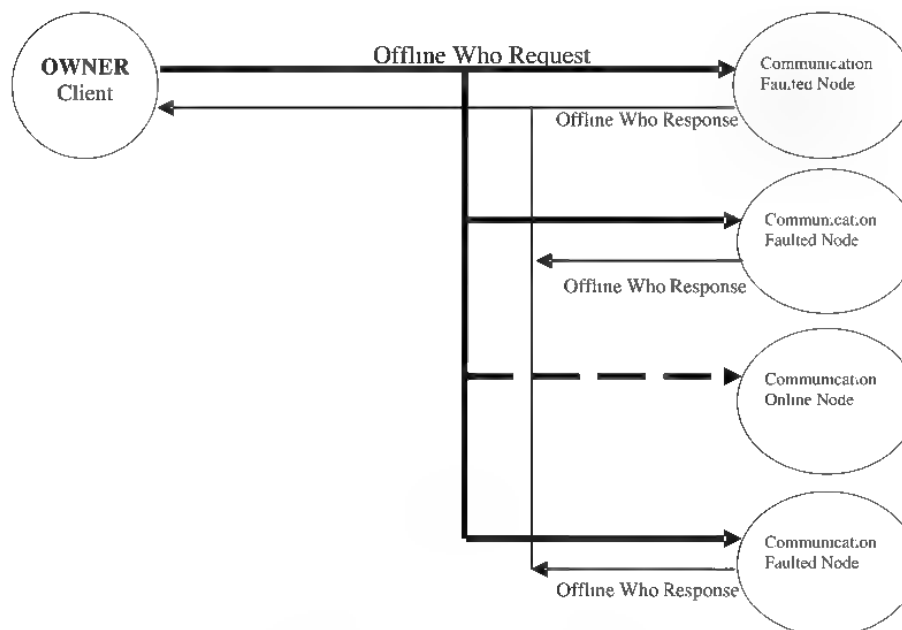
Once a client (tool) is in the Owner of the *Offline Connection Set*, it may produce *Communication Faulted Request* messages to all faulted nodes. Any node supporting the *Offline Connection Set* shall service the *Communication Faulted Request* message ONLY while in the *Communication Faulted* state.



### 2-11.3 Communication Faulted Messages

All nodes in the Communication Faulted state, which support the faulted node recovery mechanism, shall receive the Communication Faulted Request Messages produced at Group 4 Message ID 2D. When appropriate, Communication Faulted nodes shall produce a Communication Faulted Response Message at Group 4 Message ID 2C.

Figure 2-11.6 Multicast Nature of the Communication Faulted Messages



Communication Faulted Request messages are consumed by all faulted nodes supporting this functionality. Depending upon the request message, any number of the faulted nodes may reply to a single request.

#### 2-11.3.1 Communication Faulted Message Protocols

To support the Communication Faulted Message protocols a node shall support the “Network Status” LED or “Combined Module/Network Status” LED, or have a suitable device-specific method of indicating network and module status externally (such as display screen or text readout). It must also have a network settable MAC ID.

The protocol of all the *Communication Faulted Request Messages* fall into two forms, Multicast Protocol and Point-to-Point Protocol.

Figure 2-11.7 Communication Faulted Request Message - Multicast Protocol

		Contents						
Byte Offset	7	6	5	4	3	2	1	0
0	Rsv'd [0]	Match	Value					
1	R/R [0]	Service						

The Multicast Protocol is used by a client to perform requests to all *Communication Faulted* node(s) to acquire their *Serial Number(s)* and *Vendor ID(s)*. Once this information is known by the client, a Point-to-Point Protocol may be used.

When a *Communication Faulted* node receives a *Multicast Communication Faulted Request Message* it shall perform the following checks to determine whether the message should be accepted and serviced;

**Multicast Match protocol**

IF *match* bit is set (1)

AND *value* equals its *MAC ID*

AND the length of the message is two (2) bytes

THEN accept and service the *Communication Faulted Request* message

**Multicast Mask protocol**

IF *match* bit is reset (0)

AND *value* (logically AND'd with its *MAC ID*) equals its *MAC ID*

AND the length of the message is two (2) bytes

THEN accept and service the *Communication Faulted Request* message

The Multicast Mask protocol (logical AND) is used for dialogs where a range of MAC ID's are requested to respond. The following table indicates some, but not all, of the sixty-four possible mask values.

**Table 2-11.1 Addresses reporting based upon mask**

5	4	3	2	1	0	Faulted nodes with
1	1	1	1	1	1	MAC ID < 64
0	1	1	1	1	1	MAC ID < 32
0	0	1	1	1	1	MAC ID < 16
0	0	0	1	1	1	MAC ID < 8
0	0	0	0	1	1	MAC ID < 4
0	0	0	0	0	1	MAC ID < 2

The *Multicast Mask* protocol is used to reduce the number of messages, which must be sent on a subnet to determine if, and at what MAC ID, a *Communication Faulted* node resides.

The Point-to-Point Protocol follows:

Figure 2-11.8 Communication Faulted Request Message - Point-to-Point Protocol

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Reserved (Service determines usage)							
1	R/R [0]	Service						
2	Vendor ID						Low byte	
3							High byte	
4	Serial Number						Low byte	
5								
6								
7							High byte	

When a *Communication Faulted* node receives a *Point-to-Point Communication Faulted Request Message* it shall perform the following checks to determine whether the message should be accepted and serviced;

*Point-to-Point Match protocol*

IF the length of the message is equal to eight (8) bytes  
 AND the *Serial Number* and *Vendor ID* match its own  
 THEN accept and service the *Communication Faulted Request* message

The following services are currently defined for the *Communication Faulted Request* messages.

**Identify Communication Faulted Request Message - Multicast Protocol:**

Used when the client is attempting to identify the existence of any *Communication Faulted* nodes, which support the *Offline Connection Set* feature, on a subnet.

**Identify Communication Faulted Request Message - Point-to-Point Protocol:**

Used when the client detects multiple *Communication Faulted* nodes on a subnet, and wishes to visually identify a specific *Communication Faulted* node.

**Who Communication Faulted Request Message:**

Used when the client has detected *Communication Faulted* node(s) on a subnet, and wishes to acquire their *Serial Number* and *Vendor ID*.

**Change MACID Communication Faulted Request Message:**

Used when the client has detected a specific *Communication Faulted* node and wishes to change its MAC ID and have it attempt to come *On-line* at a specified MAC ID.

A device that implements the *Communication Faulted Message* Protocols shall implement support for all four of the above *Communication Faulted Message* protocols.

#### 2-11.4 Identify Communication Faulted Messages

The Client may solicit an Identify Communication Faulted Response Message from either;

1. ALL Communication Faulted nodes,
2. Communication Faulted nodes at one or more specific MAC ID's, or
3. a Communication Faulted node with a specific Vendor ID and Serial Number.

##### 2-11.4.1 Identify Request Message - Multicast protocol

The *Identify Communication Faulted Request Message - Multicast Protocol* shall be produced at Group 4 Message ID 2D. The client produces this message when it is attempting to identify the existence of any nodes in the *Communications Faulted* state.

If the length of the *Identify Communication Faulted Request* Message is two (2) bytes long, any *Communication Faulted* node(s) servicing this message shall respond but not flash their LEDs.

**Figure 2-11.9 Identify Communication Faulted Request Message - Multicast Protocol**

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Rsv'd [0]	Match	Value					
1	R/R [0]	Identify [0x4C]						

##### 2-11.4.2 Identify Response Message - Multicast protocol

The format of the *Identify Communication Faulted Response* message is the same as the received *Identify Communication Faulted Request* message (with the match and value information the same as the information in the received request) , except it is produced at Group 4 Message ID 2C, and the R/R bit is set (1). The response message shall be initiated within 100 milliseconds, upon the receipt of the request message.

**Figure 2-11.10 Communication Faulted Identify Response Message**

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Rsv'd [0]	Match	Value					
1	R/R [1]	Identify [0x4C]						

If more than one node responds to the request, collisions will not occur on the subnet. This is a result of all Communication Faulted nodes producing the same message (Connection ID and data field). However, when more than one node replies, multiple messages may be received by the client.

### 2-11.4.3 Identify Request Message - Point-to-Point protocol

The *Identify Communication Faulted Request Message - Point-to-Point Protocol* shall be produced with Group 4 Message ID 2D. If a client (tool) detects *Communication Faulted* node(s) on a subnet, and the user wishes to visually identify a *Communication Faulted* node at a specific MAC ID, then the client (tool) shall produce the *Identify Communication Faulted Request Message - Point-to-Point Protocol*.

**Figure 2-11.11 Identify Communication Faulted Request Message - Point-to-Point Protocol**

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Rsv'd [0]	Match [0]	Value [0x3f]					
1	R/R [0]	Identify [0x4C]						
2	Vendor ID							Low byte
3								High byte
4	Serial Number							Low byte
5								
6								
7								High byte

A client shall always set “match” to 0 and “value” to the 3F. A server may ignore the contents of byte 0 (checking Vendor ID and Serial Number are sufficient for uniqueness).

To acquire the *Vendor ID* and *Serial Number* of a *Communication Faulted* node, see Section 2-11.5; *Who Communication Faulted Request Message*.

IF a node accepts an *Identify Communication Faulted Request Message - Point-to-Point Protocol* AND is in the *Communication Faulted* state, it shall respond as follows:

1. Flash its bi-color *Network StatusLED* or *Combined Module/Network Status LED*. The flash rate shall alternate the Red LED ON for 250ms, followed by the Green LED turning ON for 250 ms. If the device does not have a bi-color *Network Status LED* or *Combined Module/Network Status LED*, a suitable device-specific externally viewable indication shall be provided (via display screen, etc) to show that the *Identify* message has been accepted.
2. Produce an *Identify Communication Faulted Response Message*, within 100 milliseconds.

IF a node has accepted an *Identify Communication Faulted Request Message - Point-to-Point Protocol* AND is flashing its LED, it shall stop flashing its LED under the following conditions. Similarly, if the device does not have a bi-color *Network Status LED* or *Combined Module/Network Status LED* and the device-specific indication used to indicate that an *Identify* message had been accepted is ON, it shall be turned OFF under these conditions:

1. Another *Identify Communication Faulted Request Message - Point-to-Point Protocol* is not accepted within 500 milliseconds.
2. Another *Identify Communication Faulted Request Message - Point-to-Point Protocol* is received and NOT accepted (and another *Serial Number* or *Vendor ID*).

If a node uses a Network Status LED or Combined Module/Network Status LED for the Identify message indication, the node shall turn off the LED for a minimum of 500 milliseconds, but less than one (1) second, prior to returning to its normal operation after it stops flashing the LED.

#### 2-11.4.4 Identify Response Message - Point-to-Point protocol

The Identify Communication Faulted Response Message - Point-to-Point Protocol is the same as the Identify Communication Faulted Response Message - Multicast Protocol Response (See section 2-11.4.2).

#### 2-11.5 Who Communication Faulted Request Message

The *Who Communication Faulted Request Message* shall be produced at Group 4 Message ID 2D. The *Who Communication Faulted Request Message* is used to acquire the *Serial Number* and *Vendor ID* of a Communication Faulted node

Figure 2-11.12 Who Communication Faulted Request Message

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Rsv'd [0]	Match	Value					
1	R/R [0]	Who [0x4B]						
2	Time Delay Byte Offset [ 0-6 ]							

IF the consuming node;

1. is in the Communication Faulted state, and
2. detects the *Who Communication Faulted Request Message*, and
3. meets the *match* and *value* criteria.

THEN the Communication Faulted node

1. shall wait a duration derived from the *Time Delay Byte Offset*,
2. produce a single *Who Response Message* at the Group 4 Message ID 2C, and
3. remain in the Communication Faulted state.

The client shall consume the Duplicate MAC ID Request message produced at the Communication Faulted Response Message ID 2C, thus acquiring the Communication Faulted node's Serial Number and Vendor ID.

The accuracy of the wait period shall be dependent upon the timer resolution of the internal timer within the Communication Faulted node.

The interval the faulted node waits before responding is determined as follows:

1. get the value at the *Time Delay Byte Offset* within the *Who Communication Faulted Request* message (0-6 are valid values),
2. use this value as an offset into the *Who Response Message* shown in Figure 2-11.13, then
3. multiply the byte value at that offset times fifty milliseconds, and
4. delay the response for that period of time.

## 2-11.6 Who Communication Response

The *Who Response Message*, produced following the specified delay, is the same protocol as a Duplicate MAC Request Message, except it is produced at a Group 4 Message ID 2C.

**Figure 2-11.13 Who Response Message**

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	R/R [0]	Physical Port Number						
1	Vendor ID							Low byte
2								High byte
3	Serial Number							Low byte
4								
5								
6								High byte

## 2-11.7 Change MAC ID Communication Faulted Request Message

The purpose of this message is to modify the MAC ID of a node in the Communication Faulted state. This message shall be produced using the Group 4 Message ID 2D.

Figure 2-11.14 Change MAC ID Communication Faulted Request Message

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Rsv'd [0]		Value [New MAC ID]					
1	R/R [0]	Change MAC ID [0x4D]						
2	Vendor ID							Low byte
3								High byte
4	Serial Number							Low byte
5								
6								
7								High byte

Nodes which support the Change MACID Communication Faulted Request Message feature shall not produce a Communication Faulted Response Message.

IF the consuming node

1. is in the Communication Faulted state, AND
2. detects the *Change MACID Communication Faulted Request Message*, AND
3. verifies that the *Serial Number*, and the *Vendor ID* matches its own,

THEN the faulted node

1. changes its MAC ID to the *New MAC ID* , AND
2. enters the *Sending Duplicate MAC ID* state (see section 2-3).

If, at the new MAC ID, the node fails the Duplicate MAC ID Check the node retains the new MAC ID and reenters the faulted state.

The contents of byte zero (0), bits 6 and 7 are ignored by the faulted node. Production of a value other than zero in this location is a client error, indicating non-conformance.

## 2-12 Device Heartbeat

This section defines the protocol associated with optional Device Heartbeat message. The Device Heartbeat message is triggered by the Identity Object, which is presented in Volume 1, Chapter 5, CIP Object Library.

The Device Heartbeat Message is optional.

### 2-12.1 Device Heartbeat Message

This message broadcasts the current state of the device. This message is transmitted by a UCMM capable device as an Unconnected Response Message (Message Group 3, Message ID 5) and by a group 2 only server as an Unconnected Response Message (Message Group 2, Message ID 3).



Figure 2-12.1 Device Heartbeat Message

		Contents							
Byte Offset	7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID					}	Message Header
1	R/R [1]	Service Code [4D]							
2	Identity Object Instance ID							}	Message Body
3									
4	Device State								
5	Reserved			EV	SF	UF	DF		
6	Configuration Consistency Value								
7									

Message Header

Message Body

### 2-12.1.1 Data Frame Contents, Device Heartbeat Message

**Frag (0)/Transaction ID/MAC ID** - See section 2-7.1.1, Message Header. The Heartbeat Message is an unsolicited broadcast message; there is no destination MAC ID. The source MAC ID is specified in the Message Header. This is an exception to the general rule that the message header shall not contain the same MAC ID as the CAN identifier field.

**R/R Bit (1)** - Indicates this is a response message.

**Service Code (4D<sub>hex</sub>)** - Identifies this as a Device Heartbeat Message.

**Identity Object Instance ID** - The instance ID of the identity object producing the Device Heartbeat Message. This field shall be 2 bytes in length.

**Device State** - Attribute 8 of the associated identity object instance. If attribute 8 is not supported, Device State shall be 3 (device operational).

**EV**- Event Flag, use is to be developed. This bit is ignored by the receiver and shall be set to zero by the transmitter.

**SF**- System Fault, - a fault in the device caused by bus interaction (e.g. connection timeout). This flag is set when a system fault is present.

**UF**- User Fault, - a fault in the device caused by user interaction. This flag is set when a user fault is present. The conditions under which this is set are vendor specific.

**DF**- Device Fault, - an internal fault in the device not caused by user or bus interaction (e.g. hardware fault). This flag is set when a device fault is present.

**Reserved Bits** - Use is to be developed. These bits currently are ignored by the receiver and shall be set to zero by the transmitter.

**Configuration Consistency Value** - Attribute 9 of the associated identity object instance. If attribute 9 is not supported Configuration Consistency Value shall be zero.

## 2-13 Device Shutdown Message

This section defines the protocol associated with optional Device Shutdown message. The Device Shutdown message is produced by a device when it transitions to the offline state.

The Device Shutdown Message is optional.

### 2-13.1 Device Shutdown Message

This message broadcasts the transition of a device to the offline or non-existent state. This message is transmitted by a UCMM capable device as an Unconnected Response Message (Message Group 3, Message ID 5) and by a group 2 only server as an Unconnected Response Message (Message Group 2, Message ID 3).

Figure 2-13.1 Device Shutdown Message

		Contents						
Byte Offset	7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID					} Message Header
1	R/R [1]	Service Code [4E]						
2	Class ID							} Message Body
3								
4	Instance ID							
5								
6	Shutdown Code							
7								

Message Header

Message Body

#### 2-13.1.1 Data Frame Contents, Device Shutdown Message

**Frag (0)/Transaction ID/MAC ID** - See section 2-7.1.1, Message Header. The Shutdown Message is an unsolicited broadcast message, there is no destination MAC ID. The source MAC ID is specified in the Message Header. This is an exception to the general rule that the message header shall not contain the same MAC ID as the CAN identifier field.

**R/R Bit (1)** - Indicates this is a response message.

**Service Code (4E<sub>hex</sub>)** - Identifies this as a Device Shutdown Message.

**Class ID / Instance ID** - These two values identify the object class / instance responsible for the device's transition to the offline state. If the shutdown is not caused by a specific class, Class ID shall be 0. These fields shall each be 2 bytes in length.

**Shutdown Code** - This value indicates the reason for the device's transition to the offline state.

**Table 2-13.1 Device Shutdown Message Shutdown Code Ranges**

Value	Meaning
0 <sub>hex</sub> -1FF <sub>hex</sub>	Open
200 <sub>hex</sub> -2FF <sub>hex</sub>	Vendor Specific
300 <sub>hex</sub> -4FF <sub>hex</sub>	Object Class Specific
500 <sub>hex</sub> -FFFF <sub>hex</sub>	Reserved by DeviceNet for future use

**Table 2-13.2 Device Shutdown Message “Open” Shutdown Codes**

Value	Meaning
0	Reserved
1	Operator shutdown
2	Operator reset
3	Remote shutdown
4	Remote reset
5	Internal diagnostic fault
6	Resource allocation fault
7 <sub>hex</sub> -1FF <sub>hex</sub>	Reserved

## **Volume 3: DeviceNet Adaptation of CIP**

# **Chapter 3: DeviceNet Communications**

## Contents

3-1	Introduction.....	4
3-2	DeviceNet Specific Use of Connection Object Instance Attributes .....	4
3-2.1	TransportClass_Trigger, Attribute 3 – USINT data type .....	4
3-2.2	DeviceNet_produced_connection_id, Attribute 4 – UINT data type .....	4
3-2.3	DeviceNet_consumed_connection_id, Attribute 5 – UINT data type.....	5
3-2.4	DeviceNet_initial_comm_characteristics, Attribute 6 – USINT data type .....	5
3-3	Dynamic Management Of Message IDs .....	8
3-4	Creating Connections Through the Connection Object .....	12
3-5	Creating Connections Through the DeviceNet Object.....	12
3-5.1	Allocate Master/Slave Connection_Set, Service Code 0x4B.....	12
3-5.1.1	Request Service Data Field Parameters.....	13
3-5.1.2	Data Frame Contents.....	14
3-5.1.3	Success Response Service Data Field Parameters.....	14
3-5.1.4	Data Frame Contents.....	15
3-5.2	Allocate Master/Slave Connection_Set Required Server Behavior .....	15
3-5.2.1	Protocol Examples .....	19
3-5.3	Release_Master/Slave_Connection_Set, Service Code 0x4C.....	20
3-5.3.1	Request Service Data Field Parameters.....	20
3-5.3.2	Data Frame Contents.....	21
3-5.3.3	Success Response Service Data Field Parameters.. .....	22
3-5.3.4	Data Frame Contents.....	22
3-5.4	Release_Master/Slave_Connection_Set Required Server Behavior .....	22
3-5.4.1	Protocol Examples .....	23
3-5.5	Error Codes Specific to the DeviceNet Object.....	25
3-6	Predefined Master/Slave Connection Set .....	26
3-7	Predefined Master/Slave Connection Set Messages .....	28
3-8	Slave Connection Object Characteristics .....	30
3-8.1	Connection Instance IDs .....	30
3-8.2	Slave Connection Instance Attributes .....	31
3-8.3	Predefined Master/Slave Connection Instance Behavior .....	37
3-8.4	Connection Instance Attribute Access Rules .....	41
3-9	Master Connection Object Characteristics .....	42
3-10	Bit-Strobe Command/Response Messages.....	42
3-10.1	Bit–Strobe Command Message.....	42
3-10.2	Bit–Strobe Response Message .....	44
3-10.3	Bit–Strobe Message Characteristics.. .....	44
3-10.4	Bit Strobe Example Application .....	45
3-11	Poll Command/Response Messages.....	51
3-11.1	Poll Command Message.....	51
3-11.2	Poll Response Message.....	51
3-11.3	Poll Message Characteristics.... .....	52
3-11.4	Poll Example Application .....	53
3-12	Multicast poll command/response messages.....	59
3-12.1	Multicast Poll Command Message.....	59
3-12.2	Multicast Poll Response Message .....	60
3-12.3	Multicast Poll Message Characteristics.....	60
3-12.4	Multicast Poll Example Application .....	62
3-13	Change of State/Cyclic Messages .....	68
3-13.1	Change of State/Cyclic Messages .....	71
3-13.2	Master’s Change of State/Cyclic Message.....	71
3-13.3	Slave’s Change of State/Cyclic Acknowledge Message .....	71
3-13.4	Slave’s Change of State/Cyclic Message .....	71
3-13.5	Master’s Change of State/Cyclic Acknowledge Message ... .....	72

3-13.6	Change of State/Cyclic Message Characteristics .....	72
3-13.7	Change of State/Cyclic Example Application.....	73
3-14	Group 2 Only Devices.....	79
3-14.1	Limitations of Group 2 Only Devices .....	83
3-15	Using the Predefined Master/Slave Connection Set. ....	83
3-15.1	Becoming a Master Using the Predefined Master/Slave Connection Set.....	85
3-15.2	Group 2 Only Client Responsibilities .....	89
3-15.3	Forwarding the Message .....	91
3-15.4	Possible Group 2 Only Error Scenarios .....	95
3-15.4.1	Loss of Group 2 Only Client .....	95
3-15.4.2	Duplicate Attempts to Become Group 2 Only Client....	96
3-16	Bit–Strobe and Poll filter Requirements .....	98
3-16.1	Master Device Filtering.....	98
3-16.2	Slave Device Filtering (Group 2 Only): Bit–Strobe.....	99
3-16.3	Slave Device Filtering (Group 2 Only) - Poll .....	100

## 3-1 Introduction

The CIP Communication Objects manage and provide the run-time exchange of messages. The *Services*, *Attributes*, and *Behaviors* associated with the Communication Objects are detailed in Volume 1, Chapter 3.

**Important:** It is not the intent of the following sections to specify any particular internal implementation.

The Communication Object Classes are defined by describing:

- Object Class Attributes
- Object Class Services
- Object Instance Attributes
- Object Instance Services
- Object Instance Behavior

Each CIP connection is represented by a Connection Object (Class code 0x05). The creation of this communication object resource can be done in one of two ways in DeviceNet. The two methods are:

- Use of the "Create" service (Service code 0x08) for the Connection Object
- Use of the "Allocate Master/Slave Connection Set" service (Service code 0x4B) for the DeviceNet Object

## 3-2 DeviceNet Specific Use of Connection Object Instance Attributes

The following sections describe DeviceNet-specific attributes of the Connection object. See Volume 1, Chapter 3 for the definition of the Connection Object Class (Class 0x05).

### 3-2.1 TransportClass\_Trigger, Attribute 3 – USINT data type

On DeviceNet, transport classes 2 and 3 do not prepend a 16-bit sequence count as described throughout Volume 1, Chapter 3-4.3.3.

### 3-2.2 DeviceNet\_produced\_connection\_id, Attribute 4 – UINT data type

Contains the DeviceNet Connection ID to be associated with transmissions sent across this connection (if any). This is the value that will be specified in the CAN Identifier Field when this Connection transmits. Refer to Chapter 2-2 for a description of how DeviceNet uses the CAN Identifier Field. This value is loaded directly into the associated Link Producer's **connection\_id** attribute. The following values are defined:

**Table 3-2.1** Values defined for the **produced\_connection\_id** attribute

Value	Meaning
0 - 7F0 <sub>hex</sub>	The value to be placed in the CAN Identifier Field when this Connection transmits
800 <sub>hex</sub> - FFFE <sub>hex</sub>	Reserved by CIP
FFFF <sub>hex</sub>	Default value assigned to this attribute within an I/O Connection. This attribute will retain this value if this Connection instance is not producing any data (consumer only).

### 3-2.3 DeviceNet\_consumed\_connection\_id, Attribute 5 – UINT data type

Contains the Connection ID, which identifies messages to be received across this connection (if any). This is the CAN Identifier Field value that is associated with messages this Connection Object receives. Refer to Chapter 2-2 for a description of how DeviceNet uses the CAN Identifier Field. This value is loaded directly into the associated Link Consumer's **connection\_id** attribute. The following values are defined:

**Table 3-2.2 Values defined for the consumed\_connection\_id attribute**

Value	Meaning
0 - 7F0 <sub>hex</sub>	The value that identifies messages to be consumed. This will be specified in the CAN Identifier Field of messages that are to be consumed.
800 <sub>hex</sub> - FFFE <sub>hex</sub>	Reserved by CIP
FFFF <sub>hex</sub>	Default value assigned to this attribute within an I/O Connection. This attribute will retain this value if this Connection Instance is not consuming any data (producer only)

### 3-2.4 DeviceNet\_initial\_comm\_characteristics, Attribute 6 – USINT data type

Defines the Message Group(s) across which productions and consumptions associated with this Connection occur. This attribute is required for a DeviceNet subnet. Other subnet types shall not use this attribute.

This byte is divided into two nibbles.

**Figure 3-2.1 DeviceNet\_initial\_comm\_characteristics Attribute Format**

7	6	5	4	3	2	1	0
Initial Production Characteristics				Initial Consumption Characteristics			

The following table lists the values that are possible within the *Initial Production Characteristics* nibble (upper nibble) of the **DeviceNet\_initial\_comm\_characteristics** attribute.



**Table 3-2.3 Values for the Initial Production Characteristics Nibble**

Value	Meaning	
0	Produce across Message Group 1	The production associated with this Connection is to take place across Message Group 1. The producing module generates the Connection ID value and loads it into the Connection Object's DeviceNet_produced_connection_id attribute. The producing module allocates a Message ID from its Group 1 Message ID pool and combines this with its Source MAC ID to generate the Connection ID. The numerically <u>lowest</u> available Group 1 Message ID is to be used in generating the DeviceNet_produced_connection_id attribute value. This value must also be loaded into the corresponding DeviceNet_consumed_connection_id attribute(s) associated with the consuming Connection Object(s).
1	Produce across Message Group 2 (Destination)	The production associated with this Connection is to take place across Message Group 2. Additionally, the intended recipient's MAC ID (Destination MAC ID) is to be placed within the MAC ID component of the Group 2 Identifier Field. <u>In this case, the consuming module generates the Connection ID value to be associated with transmissions across this connection.</u> When the consuming module has generated this value and loaded it into the appropriate Connection Object's DeviceNet_consumed_connection_id attribute, it can be read and subsequently loaded into the producing Connection Object's DeviceNet_produced_connection_id attribute.
2	Produce across Message Group 2 (Source)	The production associated with this Connection is to take place across Message Group 2. In addition, the producing module's MAC ID (Source MAC ID) is to be placed within the MAC ID component of the Group 2 Identifier. In this case, the producing module generates the Connection ID value and loads it into the Connection Object's DeviceNet_produced_connection_id attribute. The numerically <u>lowest</u> available Group 2 Message ID is to be used in generating the DeviceNet_produced_connection_id attribute value. This value must also be loaded into the corresponding DeviceNet_consumed_connection_id attribute(s) associated with the consuming Connection Object(s).
3	Produce across Message Group 3	The production associated with this Connection is to take place across Message Group 3. The producing module generates the Connection ID value and loads it into the Connection Object's DeviceNet_produced_connection_id attribute. The producing module allocates a Message ID from its Group 3 Message ID pool and combines this with its Source MAC ID to generate the Connection ID. The numerically <u>lowest</u> available Group 3 Message ID is to be used in generating the DeviceNet_produced_connection_id attribute value. This value must also be loaded into the corresponding DeviceNet_consumed_connection_id attribute(s) associated with the consuming Connection Object(s).
4 E	Reserved	
F	Default value	The default value assigned to the DeviceNet Initial Production Characteristics nibble within an I/O Connection. Note that if this is a <i>consuming only</i> I/O Connection, then the default value remains in this nibble. Explicit Messaging Connection Objects automatically configure this attribute when the Connection is established.

Table 3-2.4 lists the possible values within the *Initial Consumption Characteristics* nibble (lower nibble) of the **DeviceNet\_initial\_comm\_characteristics** attribute.

**Table 3-2.4 Values for the Initial Consumption Characteristics Nibble**

Value	Meaning	
0	Consume a Group 1 Message	The message to be consumed will be transmitted across Message Group 1. The producing module generates the Connection ID value. This value must be loaded into the DeviceNet_consumed_connection_id attribute associated with the consuming Connection Object(s).
1	Consume a Group 2 Message (Destination)	The message to be consumed will be transmitted across Message Group 2. The intended recipient's MAC ID (Destination MAC ID) is specified within the Group 2 Identifier. <u>The consuming module generates the Connection ID value and loads it into the DeviceNet_consumed_connection_id attribute associated with this Connection Object.</u> The numerically lowest available Group 2 Message ID is to be used in generating the DeviceNet_consumed_connection_id attribute value. This value must be loaded into the producing Connection Object's DeviceNet_produced_connection_id attribute.
2	Consume a Group 2 Message (Source)	The message to be consumed will be transmitted across Message Group 2. The transmitting module's MAC ID (Source MAC ID) is specified within the Group 2 Identifier. In this case, the producing module generates the Connection ID value and loads it into the Connection Object's the DeviceNet_produced_connection_id attribute. This value must be loaded into the DeviceNet_consumed_connection_id attribute associated with the consuming Connection Object(s).
3	Consume a Group 3 Message	The message to be consumed will be transmitted as a Group 3 Message. The producing module generates the Connection ID value. The Connection ID value must be loaded into this Connection Object's DeviceNet_consumed_connection_id attribute.
4 - E	Reserved by CIP	
F	Default value	The default value assigned to the DeviceNet Initial Consumption Characteristics nibble within an I/O Connection. Note that if this is a <i>producing only</i> I/O Connection, then the default value remains in this nibble. Explicit Messaging Connections automatically configure this attribute when the Connection is established.

**Important:** The module that generates a Connection ID must guarantee that it does not allocate the Message ID/MAC ID pair in such a way that two separate modules are capable of transmitting identical bit patterns within the Identifier Field. Refer to section 3-3 for further details.

### 3-3 Dynamic Management Of Message IDs

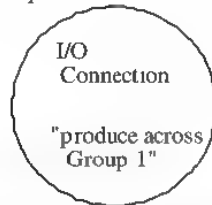
Dynamically establishing both I/O and Explicit Messaging Connections requires the end-points go through an internal Message ID allocation process. For example:

**Figure 3-3.1 Dynamic Management of Message IDs**

**1. Current state of the "Group 1 Message ID Allocation Table"**

Message ID 0	Allocated
Message ID 1	Allocated
Message ID 2	Available
Message ID 3	Available
.....	
Message ID 0F	Available

**2. An I/O Connection is created & configured which requests that a new production take place across Group 1**



**3. The next available Group 1 Message ID is allocated for use within the I/O Connection**

Message ID 0	Allocated
Message ID 1	Allocated
Message ID 2	Allocated
Message ID 3	Available
.....	
Message ID 0F	Available

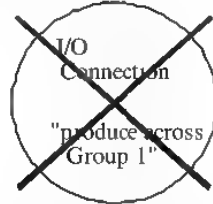
Additionally, devices which implement fully dynamic Connection creation/deletion capabilities will also have to implement logic which marks a previously allocated Message ID as available when that Message ID is no longer in use. For example:

**Figure 3-3.2 Dynamic Management of Message IDs**

**1. Current state of the "Group 1 Message ID Allocation Table"**

Message ID 0	Allocated
Message ID 1	Allocated
Message ID 2	Allocated
Message ID 3	Available
.....	
Message ID 0F	Available

**2. The I/O Connection is deleted**



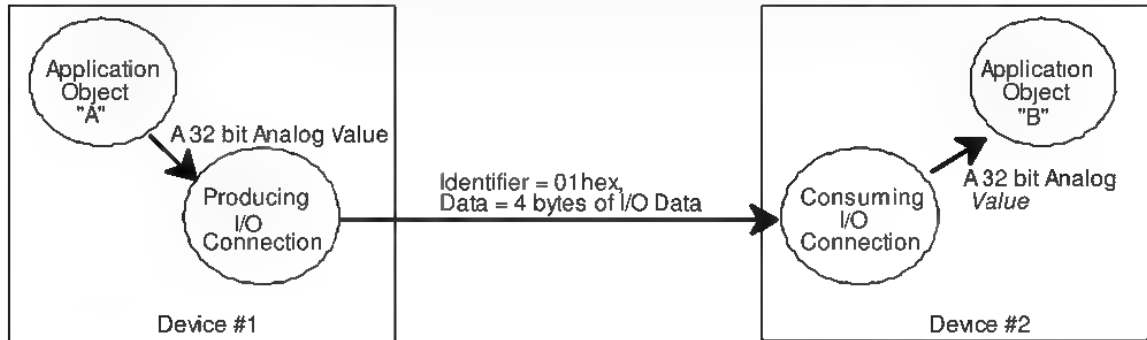
**3. Group 1 Message ID value 2 is now available again.**

Message ID 0	Allocated
Message ID 1	Allocated
Message ID 2	Available
Message ID 3	Available
.....	
Message ID 0F	Available

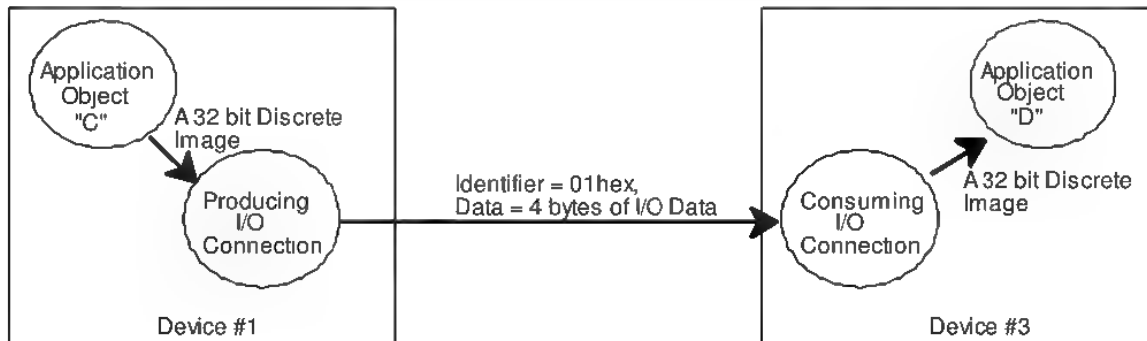
The act of re-using a previously allocated Message ID for a different purpose gives rise to many issues. For example:

**Figure 3-3.3 Message ID Reuse example**

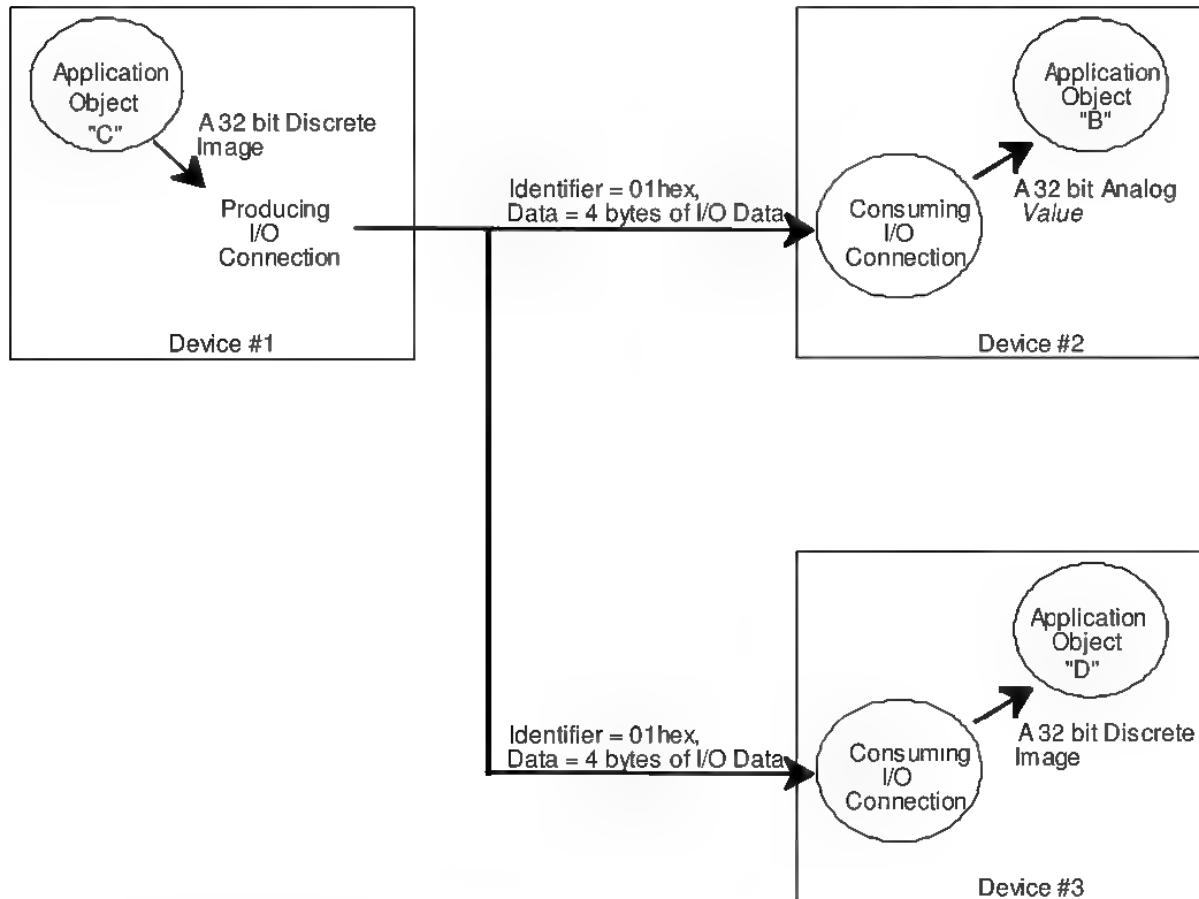
1. Assume the following configuration where the Producing I/O Connection in Device #1 is Analog Value identified by Identifier Field value 01hex. The Consuming I/O Connection is Identifier 01hex and when a consumption occurs it hands the data up to Application Object 'B'. Application Object 'B' assumes it is receiving an Analog Value and processes accordingly



2. Assume that the Producing I/O Connection is Deleted and the associated Message ID is marked as available. Assume a new Producing I/O Connection is created and configured such that it re-uses this Message ID and generates the Identifier 01hex which is now associated with a 32 bit Discrete Device #3 is configured to consume this message as illustrated.



3 There is a problem in that Application Object B within Device #2 processes a received I/O Message as a 32 bit Analog Value. The associated Consuming I/O Connection within Device #2 is still active and screening for Message ID 01hex. Application Object B in Device #2 will now receive what it believes to be an Analog Value but it is actually a 32 bit Discrete Image from a totally different source Application Object!



Not one single approach can resolve these Connection ID-related error scenarios in all situations/systems. To reduce the likelihood of Connection ID related error scenarios, the following implementation guidelines must be followed:

The allocation process must make every attempt to ensure that no two modules are capable of transmitting an identical bit pattern within the Connection Identifier Field.

If possible, the deallocation process must ensure that all end-points of a Connection have timed out prior to re-using a Message ID.

To further refine guideline #2, apply the following logic **when the decision has been made to mark a previously allocated Message ID as available:**

- If the Message ID is associated with a Connection that activates an Inactivity/Watchdog Timer, then a new Inactivity/Watchdog timer is activated. Upon expiration of this timer, the Message ID is marked as available.

- If the Message ID is associated with a Connection that does not activate an Inactivity/Watchdog Timer, then the assumption is made that this was taken into consideration when the Connection was established and the Message ID can be immediately marked as available.

**Important:** Realize that for a Connection using Transport Class 0 or a Connection whose `expected_packet_rate` attribute has been set to zero (0), guideline #2 cannot be met using logic based on that Connection Object alone. For example:

- If the Server end-point of a Transport Class 0 Connection experiences an Inactivity/Watchdog timeout, it cannot know the state of the Client based solely on this Connection. The Client could have just missed transmitting the message in a timely fashion and could still think the Connection is operating normally.
- If the Client end-point of a Connection whose `expected_packet_rate` has been set to zero (0) is deleted for some reason, the Server end-point(s) could still be active.

For the types of connections discussed in the bullet list above, use caution when performing tasks that will result in the deallocation and possible re-use of the associated Message ID(s). (i.e., configuring the `watchdog_timeout_action` attribute to *Auto Delete*, manually Deleting a Connection Object via the transmission of the Delete service)

### 3-4 Creating Connections Through the Connection Object

When the subnet defines that connections are created through the Connection Object, a CIP device shall support the Create service for this class. The Create service instantiates a Connection Instance with attribute values defaulted as defined by the class. The connection instance is configured through individual access to each Connection instance attribute. A separate service request (Apply\_Attributes, Service code 0x0D) is needed to transition the connection to the Established state.

### 3-5 Creating Connections Through the DeviceNet Object

When connections are created via the DeviceNet Object, the Allocate Master/Slave Connection Set (Service code 0x4B) is used to instantiate the desired connections.

#### 3-5.1 Allocate\_Master/Slave\_Connection\_Set, Service Code 0x4B

This is the Service utilized to perform the allocation of the Predefined Master/Slave Connection Set. Status Codes listed below in *italics* are defined in Volume 1, Appendix B. DeviceNet Object defined Additional Code values are presented in section. This service can be transmitted across the Group 2 Only Unconnected Explicit Request Message Port (Group 2 Message ID = 6 - see section 3-14) as well as an Explicit Messaging Connection.

The Allocate\_Master/Slave\_Connection\_Set service bundles the following general steps into one command:

- Connection Object Create
- Connection Object Configuration

### 3-5.1.1 Request Service Data Field Parameters

The following information is specified within the Service Data Field of an Allocate\_Master/Slave\_Connection\_Set request.

**Table 3-5.1 Allocate\_Master/Slave\_Connection\_Set Request Service Data Field Parameters**

Name	Data Type	Description of Service Parameter
Allocation Choice	BYTE	Indicates which Connections from the Predefined Master/Slave Connection Set are to be allocated/configured for use by the Master
Allocator's MAC ID	USINT	Contains the MAC ID associated with the module requesting the allocation. This may seem redundant in that the MAC ID is also specified in the DeviceNet Explicit Message Header, but due to the fact that a Group 2 Only Client may be receiving requests from another device and forwarding them on to a Group 2 Only Server (see section 3-14), the MAC ID in the Message Header may not accurately reflect who is requesting the allocation.

The Allocation **Choice** parameter is specified within a single byte. Each bit denotes an Explicit Message and/or I/O Connection(s) from the Predefined Master/Slave Connection Set that are to be allocated, or in the case of Acknowledge Suppression, a command. If the bit is set to one (1), then a request is being made to allocate that particular Connection. If a bit is set to zero (0), then the requester does not want to allocate that Connection.

**Table 3-5.2 Allocation Choice Byte Contents**

7	6	5	4	3	2	1	0
Reserved	Acknowledge Suppression	Cyclic	Change of State	Multicast Polling	Bit Strobed	Polled	Explicit Message

A Slave should check to make sure no *Reserved* bits are set when the Allocate\_Master/Slave\_Connection\_Set request is received. Reserved bits may become valid choices in the future; a present day server could not support the request, so it must return an error. A value of 00 is also invalid.

The following illustration depicts the format of this Explicit Message on DeviceNet.

**Figure 3-5.1 Allocate\_Master/Slave\_Connection\_Set Request Message**

		Contents							
Byte Offset		7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID						
1	R/R [0]	Service Code [4B]							
	Class ID [3]								
	Instance ID [01]								
	Allocation Choice								
	0	0	Allocator's MAC ID						



### 3-5.1.2 Data Frame Contents

**Frag (0)/Transaction ID/MAC ID** - See Chapter 2-7.1.1, Message Header.

**R/R Bit (0)** - Indicates this is a request message.

**Service Code (4B<sub>hex</sub>)** - Identifies this as an Allocate Master/Slave Connection Set service.

**Class ID** - Defines the object class towards which this request is directed. Since this message is always directed to the DeviceNet Object, this will always be a three (3). When this message is transmitted across a Explicit Messaging Connection, the Class ID will be specified in either an 8 or 16 bit integer field based on the Message Body Format negotiated when the Messaging Connection was established. When this message is transmitted across the Group 2 Only Unconnected Explicit Request Message Port (Group 2 Message ID = 6), the Class ID value is specified within an 8 bit integer field.

**Instance ID** - Defines the particular instance within the object class towards which this request is directed. When this message is transmitted across an Explicit Messaging Connection, the Instance ID will be specified in either an 8-bit or 16-bit integer field based on the Message Body Format negotiated when the Messaging Connection was established. When this message is transmitted across the Group 2 Only Unconnected Explicit Request Message Port (Group 2 Message ID = 6), the Instance ID value is specified within an 8-bit integer field. Since there is one, and only one, DeviceNet Object per physical attachment to DeviceNet, the Instance ID must be set to 01.

**Allocation Choice** - Specified within the byte following the Instance ID field.

**Allocator's MAC ID** - Specified within the byte following the Allocation Choice field.

### 3-5.1.3 Success Response Service Data Field Parameters

The following information is specified within the Service Data Field of a successful Allocate\_Master/Slave\_Connection\_Set response.

**Table 3-5.3 Allocate\_Master/Slave\_Connection Set Response Parameters**

Name	Data Type	Description of Parameter
Message Body Format	Defined below	This is semantically equivalent to the <i>Actual Message Body Format</i> parameter returned with a DeviceNet Open Explicit Message Connection Response Message. This argument is significant when the Allocate_Master/Slave_Connection_Set request was received across the Group 2 Only Unconnected Explicit Request Message Port (UCMM incapable device). It indicates the Message Body Format associated with subsequent messages transmitted across the Explicit Messaging Connection within the Predefined Master/Slave Connection Set (Group 2 Message ID 4). If the Allocate_Master/Slave_Connection_Set request was received across an Explicit Messaging Connection within a UCMM capable device, then this parameter is set to the Actual Message Body Format associated with that Explicit Messaging Connection.

The following illustration depicts the format of a successful response to the Allocate\_Master/Slave\_Connection\_Set request on DeviceNet.

Figure 3-5.2 Success Response to Allocate\_Master/Slave\_Connection\_Set Request

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID					
1	R/R [1]	Service Code [4B]						
2	Reserved (all bits=0)				Message Body Format			

### 3-5.1.4 Data Frame Contents

**Frag (0)/Transaction ID/MAC ID** - See Chapter 2-7.1.1, Message Header.

**R/R Bit (1)** - Indicates this is a response message.

**Service Code (4B<sub>hex</sub>)** - Identifies this as an Allocate\_Master/Slave\_Connection\_Set service.

**Reserved Bits** - Use is to be developed. These bits are to be ignored by the receiver of the response and should be set to zero by the transmitter of the response message.

**Message Body Format** - See Table 3-5.3.

UCMM capable servers will send the success response across a previously established Explicit Messaging Connection. The value of the Message Body Format parameter should be the same as that returned when the Explicit Message connection was opened. In this case, the Client should shall ignore this parameter.

## 3-5.2 Allocate\_Master/Slave\_Connection\_Set Required Server Behavior

1. If the receiving device does not support the Predefined Master/Slave Connection Set, then an Error Response is returned. The General Error Code within the Error Response is set to 08 to indicate *Service not supported*.
2. The receiving device (Slave) validates the **Allocator's MAC ID** parameter within the request as follows:
  - If the Predefined Master/Slave Connection Set is allocated and this request is not from the current Master (DeviceNet Object's MAC ID attribute is NOT equal to the Allocator's MAC ID parameter), then the Slave returns an Error. The General Error Code within the Error Response is set to 0C<sub>hex</sub> (*Object cannot perform service in its current mode/state*), with the Additional Code set to an Object Specific value of 01 (see section 3-5.5).
  - If the Predefined Master/Slave Connection Set is not allocated, then the Slave validates the Allocation Choice parameter as specified in number 3.
  - If the Predefined Master/Slave Connection Set is allocated and this request is from the current Master (Master's MACID portion of the Allocation\_Information attribute is equal to the Allocator's MAC ID parameter), then the Slave validates the Allocation Choice parameter as specified in number

3. The Slave validates the **Allocation Choice** parameter within the request. If the Slave does not support one of the Connections specified in the **Allocation Choice** argument (including the Reserved bits), then an Error Response is returned. The General Error Code within the Error Response is set to 02 (*Resource Unavailable*), with the Additional Code set to an Object Specific value of 02.

If **any** of the Connection(s) being requested are supported by this Slave and have already been allocated to the Master denoted by the **Allocator's MAC ID** argument, the Slave returns an Error Response with the General Error Code set to 0B (*Already In Requested Mode/State*), with the Additional Code set to an Object Specific value of 02. The Master should consider sending a Release request as it may be out-of-sync with the Slave. The exception to this is when the requested I/O connection is in the Timed-Out state. In this case the Slave reallocates the I/O connection, sending it back to the Configuring State.

If the Allocation Choice byte has no bits set (all bits are zero) the Slave returns an Error Response with the General Error Code set to 09 (*Invalid Attribute Value*), with the Additional Code set to an Object Specific value of 02.

If a resource that is required for use with the requested Connections is not available, then an Error Response is returned with the General Error Code set to 02 (*Resource Unavailable*), and the Additional Code set to an Object Specific value of 04.

The Change of State and Cyclic allocation choices are mutually exclusive. If an Allocation Request would result in both the cyclic and change of state bits being set, an Error Response is returned. The General Error Code within the Error Response is set to 09h (Invalid Attribute Value) with an Additional Code of 02h. The Bit Strobed and Polled choices can coexist with either the Change of State or Cyclic choice.

If a master has allocated the Change of State/Cyclic connection set, and a subsequent allocation request is received with the Polled allocation bit set, an error is returned. The error will indicate Error Code 02 (Resource Unavailable) with the Additional Code set to an object specific value of 04.

If either the Change of State and/or Cyclic connection sets are supported, the Acknowledge Suppression feature shall also be supported.

If the Allocation Choice byte has the Acknowledge Suppression bit set and neither the Change of State nor the Cyclic are set, an error response is returned. The General Error Code within the Error Response is set to 09h (Invalid Attribute Value) with an Additional Code of 02h.

**Important:** If an error is encountered, then none of the requested connections are allocated. If this request cannot be fully serviced, then none of the requested allocations take place.

4. The Slave notes the fact that the Predefined Master/Slave Connection Set has been allocated to the MAC ID within the **Allocator's MAC ID** field by updating the DeviceNet Object's Allocation Information attribute. If necessary, the **produced\_connection\_id** and/or **consumed\_connection\_id** attributes of the Connection Object(s) can now be initialized (Note that the *Polled I/O* and Explicit Messaging Connections can be initialized prior to the allocation. *Bit-Strobed I/O* Connection needs to obtain the MAC ID of the Master prior to initializing the **consumed\_connection\_id** attribute. See section 3-6).

The Allocation Choice byte of the Allocation Information attribute indicates which Connection Objects from the Predefined Master/Slave Connection Set are active (in the Configuring, or Established state). This byte may need to be updated whenever a Master/Slave Connection Object changes state.

5. Any allocated I/O Connection(s) transition to the Configuring state. With respect to the Predefined Master/Slave Connection Objects, an implied Apply\_Attributes service accompanies a Set\_Attribute\_Single of the **expected\_packet\_rate** attribute while the connection is in the **Configuring** state. A Set\_Attribute\_Single of the **expected\_packet\_rate** attribute triggers the execution of the steps performed with an Apply\_Attributes service and causes the predefined Master/Slave I/O Connection Object to transition to the **Established** state. If an error is encountered when performing the *Apply\_Attributes* portion (see Table 3-4.20, I/O Connection State Event Matrix table in Volume 1, Chapter 3-4.7.1) of the Set\_Attribute\_Single to the **expected\_packet\_rate** attribute, then:

1. The **expected\_packet\_rate** attribute is returned to its previous value (the value present before the reception of the Set\_Attribute\_Single request).
2. The error is reported in an Error Response to the Set\_Attribute\_Single Request whose General Error Code is set to *Invalid Attribute Value* (09<sub>hex</sub>) and whose Additional Code is set to the Attribute ID of the *offending* Connection Object Attribute ID (this mirrors the behavior specified in Table 3-4.20, I/O Connection State Event Matrix table in Volume 1, Chapter 3-4.7.1 for an Apply\_Attributes Error Response).

Remember that the actual value loaded into the **expected\_packet\_rate** attribute is returned in the successful *Set\_Attribute\_Single* response.

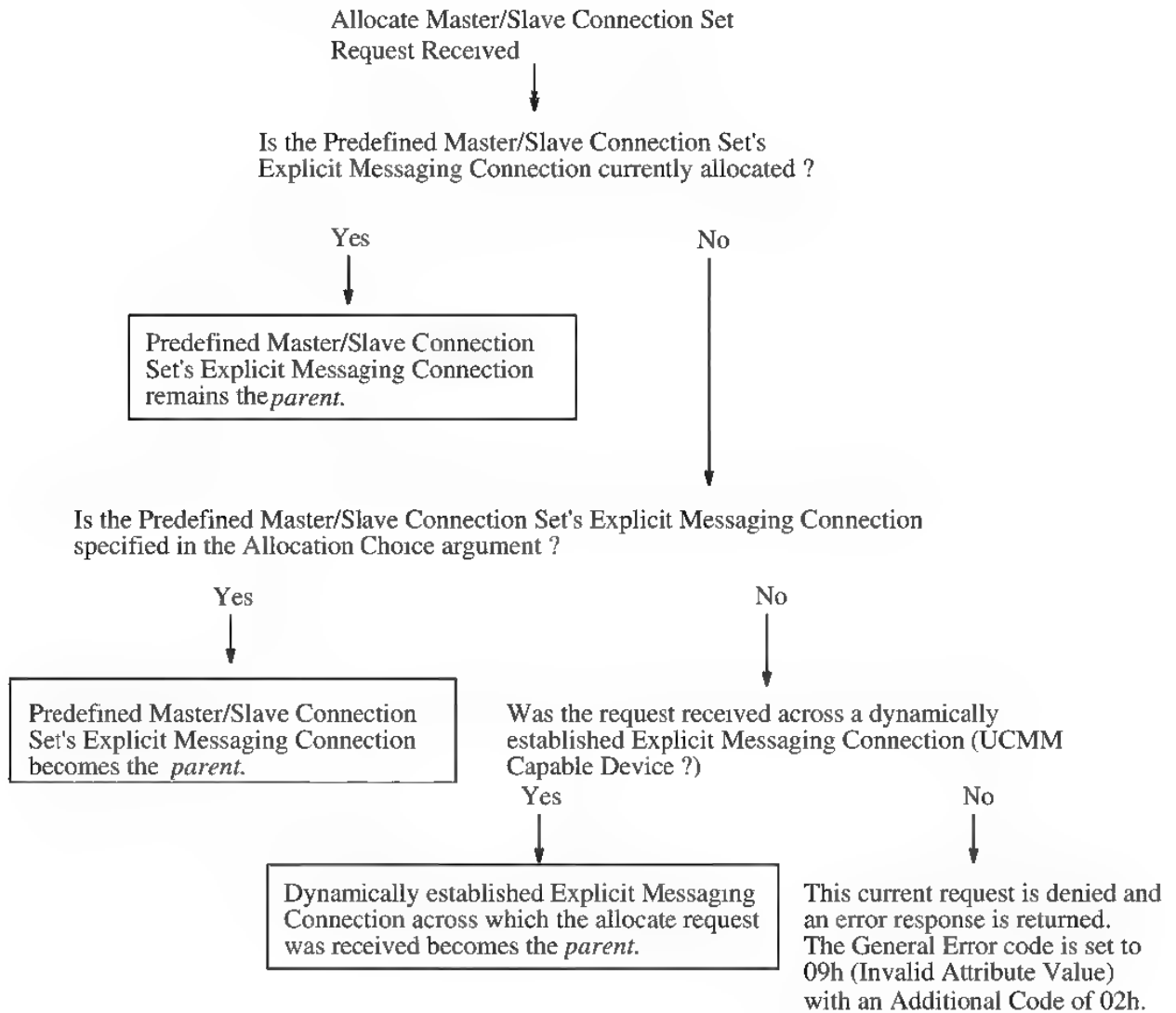
The Inactivity/Watchdog Timer functions as described in Volume 1, Chapter 3-4.5.2.

6. If allocated, the Explicit Messaging Connection transitions to the Established state. The Inactivity/Watchdog Timer is started using the *pre\_consumption* value specified in Volume 1, Chapter 3-4.5.2.

**Important:** UCMM capable devices which support the Predefined Master/Slave Connection Set must support the allocation and use of the Predefined Master/Slave Connection Set Explicit Messaging connection.

7. Either the Master/Slave Connection Set's Explicit Messaging Connection or a dynamically established Explicit Messaging Connection is the allocated I/O Connection(s) *parent* **unless they are in the Established state**. The term *parent* is used to indicate that if the Explicit Messaging Connection is deleted and none of its allocated (child) I/O Connections exist in the **Established** state, then the Predefined Master/Slave Connection Set is automatically released by the Slave. The logic describing *which* Explicit Messaging Connection is to act as the *parent* is presented in Figure 3-5.3. Note that Figure 3-5.3 assumes that all validation checks described above have been successfully performed.

Figure 3-5.3 Parent Explicit Messaging Connection Logic



**Important:** The Slave automatically releases the Predefined Master/Slave Connection Set if **all** of the following conditions are **true**:

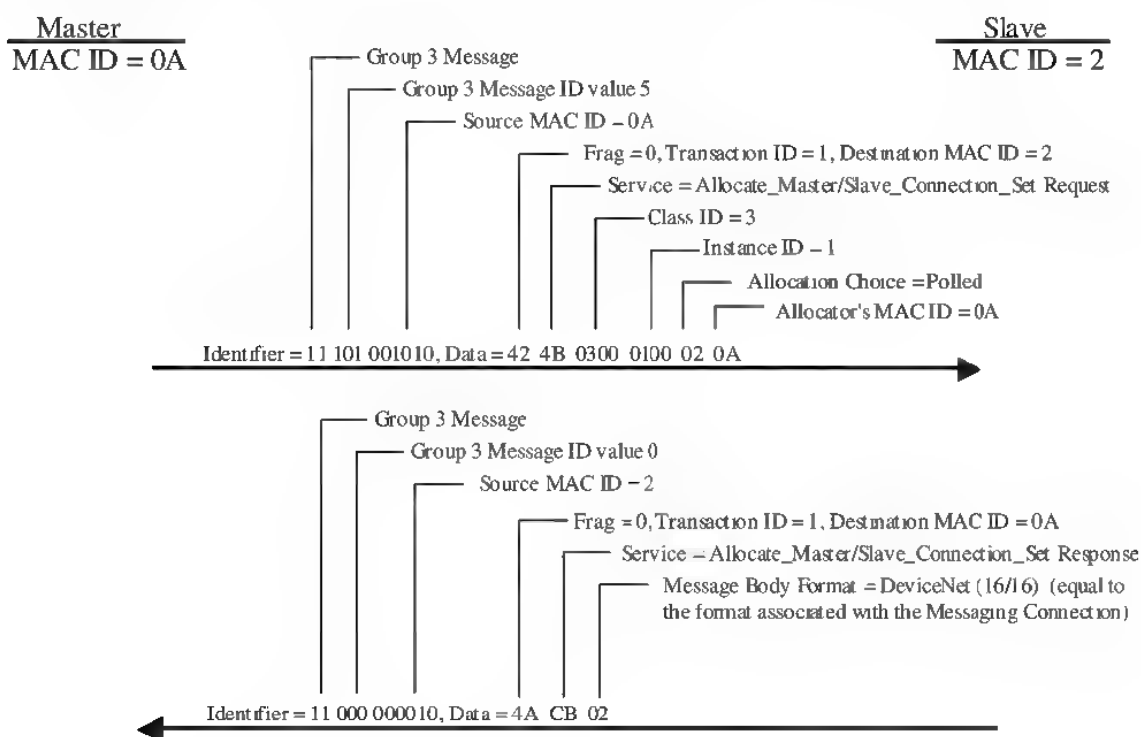
- If none of the Connection Objects associated with the Predefined Master/Slave Connection Set exist in the Established state
- If the *parent* Explicit Messaging Connection Object is not in the **Established** state

If all of the above conditions are true, then the Slave releases the Predefined Master/Slave Connection Set and all Connections return to the non-existent state. Implementers should note that for Group 2 Only Servers, a Master must allocate the Explicit Messaging Connection prior to, or along with, allocating I/O Connections.

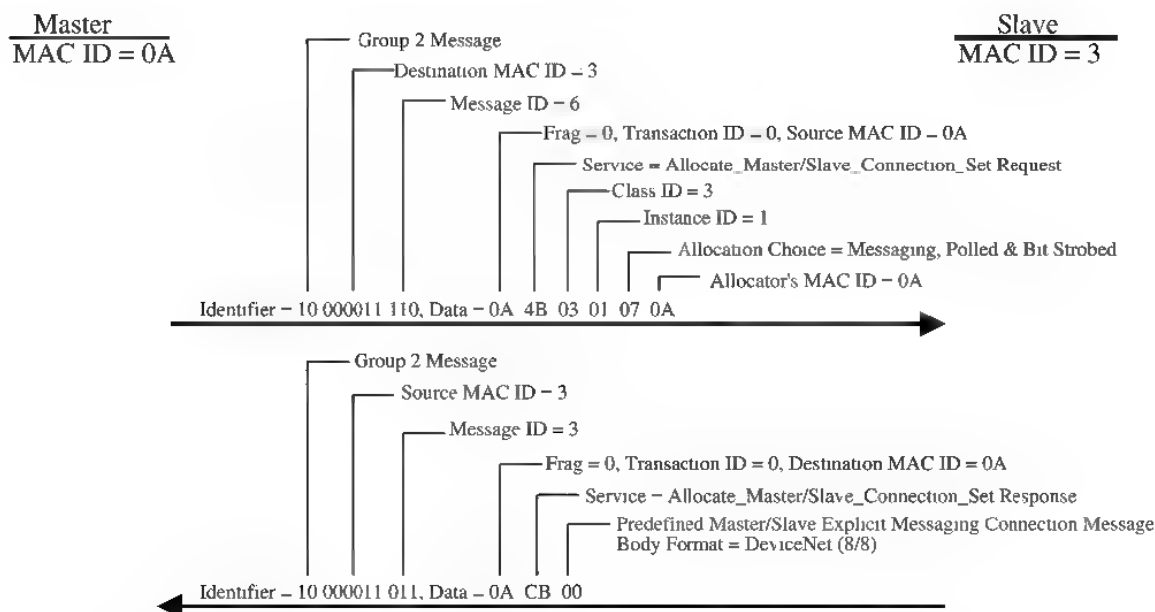
**Important:** Section 3-8 and 3-9 present the formal definition of the Connection Objects associated with the Predefined Master/Slave Connection Set.

### 3-5.2.1 Protocol Examples

The example below illustrates the successful allocation of the Predefined Master/Slave Connection Set's Polled I/O Connection. The Slave is a UCMM capable device. Assume an Explicit Messaging Connection was previously established across Group 3 and the Message Body Format is DeviceNet (16/16).



The example below illustrates the Predefined Master/Slave Connection Set being allocated from a UCMM Incapable device. This illustrates the utilization of the Group 2 Only Unconnected Explicit Message port described in section 3-14 to perform the allocation.



**Important:** The Allocate\_Master/Slave\_Connection\_Set and Release\_Master/Slave\_Connection\_Set Requests are routed to the DeviceNet Object associated with the physical network attachment across which the request is received. Since these messages can be transmitted across Explicit Messaging Connection, the Class and Instance IDs are specified in the request messages.

### 3-5.3 Release\_Master/Slave\_Connection\_Set, Service Code 0x4C

This service is used to deallocate the Predefined Master/Slave Connection Set within a Slave. Error Codes listed below in italics are defined in Volume 1, Appendix B. DeviceNet Object defined Additional Code values are presented in section 3-5.5. This service can be transmitted across the Group 2 Only Unconnected Explicit Request Message Port (Group 2 Message ID = 6 - see section 3-14) as well as an Explicit Messaging Connection.

#### 3-5.3.1 Request Service Data Field Parameters

The following information is specified within the Service Data Field of a Release\_Master/Slave\_Connection\_Set request.

**Table 3-5.4 Release Master/Slave Connection Set Request Parameters**

Name	Data Type	Description of Parameter
Release Choice	BYTE	Indicates which Predefined Master/Slave Connection's are to be "released". The process of releasing the Connection brings it back to a state whereby it can once again be allocated (back to its initial state).

The **Release Choice** parameter is specified within a single byte. Each bit denotes an Explicit Message and/or I/O Connection(s) to be released. If the bit is set to one (1), then a request is being made to release that particular Connection. If a bit is set to zero (0), then the requester does not want to release that Connection.

**Table 3-5.5 Release Choice Byte Contents**

7	6	5	4	3	2	1	0
Reserved	Note	Cyclic	Change of State	Multicast Polling	Bit Strobed	Polled	Explicit Message

**Note:** This bit shall be set to zero. The Server may optionally check this bit and return an error code response if it is non-zero.

The Server shall check to make sure no *Reserved* bits are set when the Release\_Master/Slave\_Connection\_Set request is received. Reserved bits may become valid choices in the future; a present day server could not support the request, so it shall return an error. A value of 00 is also invalid.

The following illustration depicts the format of this Explicit Message on DeviceNet.

**Figure 3-5.4 Release\_Master/Slave\_Connection Set Request Message**

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID					
1	R/R [0]	Service Code [4C]						
	Class ID [3]							
	Instance ID [01]							
	Release Choice							

### 3-5.3.2 Data Frame Contents

**Frag (0)/Transaction ID/MAC ID** - See Chapter 2-7.1.1, Message Header.

**R/R Bit (0)** - Indicates this is a request message.

**Service Code (4C<sub>hex</sub>)** - Identifies this as a Release\_Master/Slave\_Connection\_Set service.

**Class ID** - Defines the object class towards which this request is directed. Since this message is always directed to the DeviceNet Object, this will always be a three (3). When this message is transmitted across a Explicit Messaging Connection, the Class ID will be specified in either an 8 or 16 bit integer field based on the Message Body Format negotiated when the Messaging Connection was established. When this message is transmitted across the Group 2 Only Unconnected Explicit Request Message Port (Group 2 Message ID = 6), the Class ID value is specified within an 8 bit integer field.



**Instance ID** - Defines the particular instance within the object class towards which this request is directed. When this message is transmitted across an Explicit Messaging Connection, the Instance ID will be specified in either an 8 or 16 bit integer field based on the Message Body Format negotiated when the Messaging Connection was established. When this message is transmitted across the Group 2 Only Unconnected Explicit Request Message Port (Group 2 Message ID = 6), the Instance ID value is specified within an 8-bit integer field. Since there is one, and only one, DeviceNet Object per physical attachment to DeviceNet, the Instance ID must be set to 01.

**Release Choice** - Specified within the byte following the Instance ID field.

### 3-5.3.3 Success Response Service Data Field Parameters

NONE

The following illustration depicts the format of a successful response to this request on DeviceNet.

Figure 3-5.5 Release\_Master/Slave\_Connection\_Set Response Message

Byte Offset	Contents							
	7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID					
1	R/R [1]	Service Code [4C]						

### 3-5.3.4 Data Frame Contents

**Frag (0)/Transaction ID/MAC ID** - See Chapter 2-7.1.1, Message Header.

**R/R Bit (1)** - Indicates this is a response message.

**Service Code (4C<sub>hex</sub>)** - Identifies this as a Release\_Master/Slave\_Connection\_Set service.

See the DeviceNet Messaging Protocol Chapter for a detailed description of the format of an Error Response.

### 3-5.4 Release\_Master/Slave\_Connection\_Set Required Server Behavior

1. If the receiving device does not support the Predefined Master/Slave Connection Set, then an Error Response is returned. The General Error Code within the Error Response is set to 08 to indicate *Service not supported*.
2. The receiving device (Slave) validates the **Release Choice** parameter within the request. If the Slave does not support one of the Connections specified in the **Release Choice** argument (including the TBD bits), then an Error Response is returned. The General Error Code within the Error Response is set to 02 (*Resource Unavailable*), with the Additional Code set to an Object Specific value of 02.

If the Release Choice byte has no bits set (all bits are zero) the Slave returns an Error Response with the General Error Code set to 09 (*Invalid Attribute Value*), with the Additional Code set to an Object Specific value of 02.

If either the Change of State or Cyclic connection set is released, the Acknowledge Suppression “bit” is released.

**Important:** If an error is encountered, then none of the specified connections are released. If this request cannot be fully serviced, then none of the requested releases take place.

3. If one of the specified Connections is in the non-existent state, then an Error Response is returned. The General Error Code within the Error Response is set to 0B to indicate *Already In Requested Mode/State*.
4. The Slave ensures that it is in a state that allows it to discontinue use of the specified Connection(s). If this is not the case, then an Error Response is returned. The General Error Code within the Error Response is set to 0C to indicate *Cannot Perform Service In Current Mode/State*.
5. If the request is valid, then the Slave releases all resources associated with the specified Connection(s). If this results in all Predefined Master/Slave Connections being released, the Slave notes the fact that the Predefined Master/Slave Connection Set is no longer allocated by updating the Allocation Information attribute.

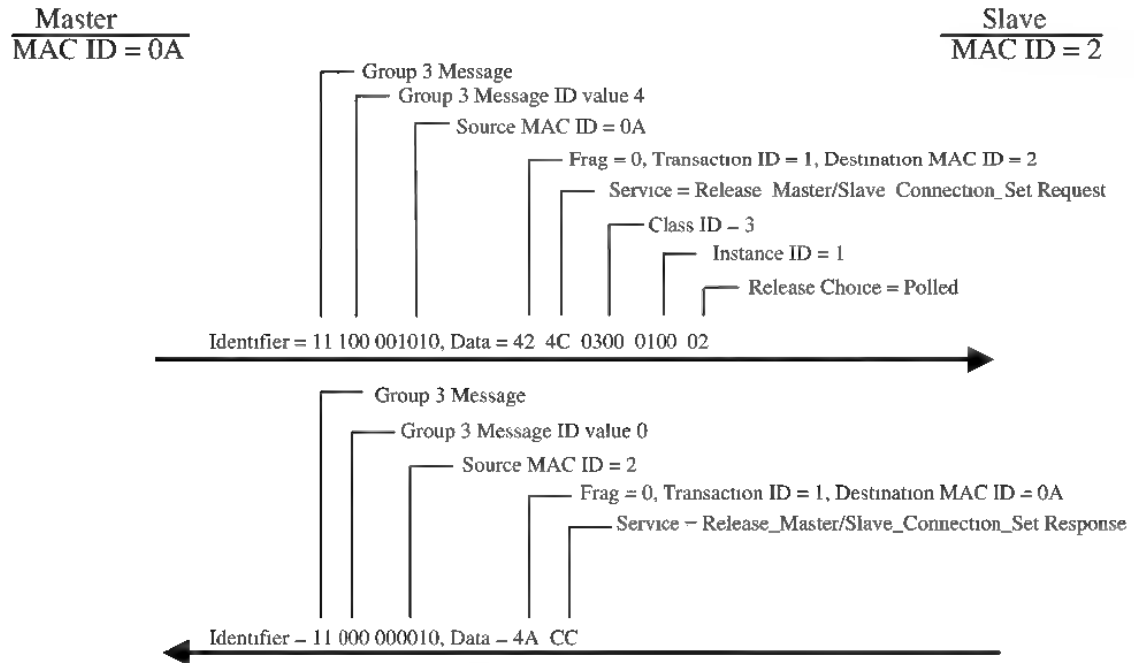
**Important:** Note that the Slave does not check to see that the release request came from its Master.

6. If this request has resulted in none of the Predefined Master/Slave Connections existing in the **Established** state, then the Slave releases the Predefined Master/Slave Connection Set and all Connections return to the non-existent state.

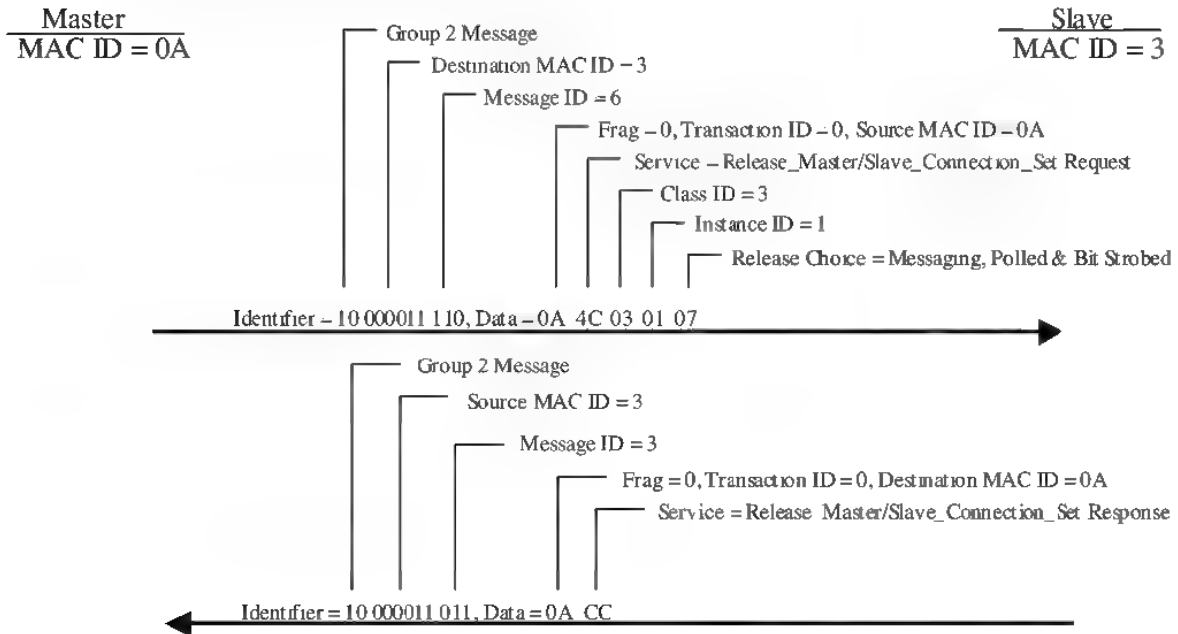
**Important:** The Allocate\_Master/Slave\_Connection\_Set and Release\_Master/Slave\_Connection\_Set Requests are routed to the DeviceNet Object associated with the physical network attachment across which the request is received. Since these messages can be transmitted across Explicit Messaging Connection, the Class and Instance IDs are specified.

### 3-5.4.1 Protocol Examples

The example below illustrates the successful release of the Predefined Master/Slave Connection Set's Poll I/O Connection. The Slave is a UCMM capable device. Assume an Explicit Messaging Connection was previously established across Group 3 and the Message Body Format is DeviceNet (16/16).



The example below illustrates the Predefined Master/Slave Connection Set's Polled I/O, Bit Strobed I/O, and Explicit Messaging Connections being released from a UCMM incapable device. This illustrates the utilization of the Group 2 Only Unconnected Explicit Message port described in section 3-14 to perform the release.



### 3-5.5 Error Codes Specific to the DeviceNet Object

The following table lists error codes specific to the DeviceNet Object. These codes would be placed in the Additional Code field within an DeviceNet Error Response message. These codes are referenced/described in detail throughout the preceding sections.

**Table 3-5.6 DeviceNet Object Specific Additional Error Codes**

Value	Meaning
01	Predefined Master/Slave Connection Set allocation conflict. This is returned when an Allocate_Master/Slave_Connection_Set request is received and the Slave has already allocated the Predefined Master/Slave Connection Set to another Master.
02	Invalid Allocation/Release Choice parameter. This is returned when an Allocate/Release Master/Slave Connection_Set request is received and.  1) The Slave does not support the choice specified in the Choice parameter. 2) The Slave was asked to Allocate/Release connection(s) already allocated/released. 3) The Allocation Choice/Release byte contained all zeros, an invalid combination of bits, or did not contain the Explicit Message Allocation Choice when required
03	A Group 2 Only Server (UCMM incapable) received a message on the Group 2 Only Unconnected Explicit Request message port that was not an Allocate or Release message.
04	Resource required for use with the Predefined Master/Slave Connection Set is not available

### 3-6 Predefined Master/Slave Connection Set

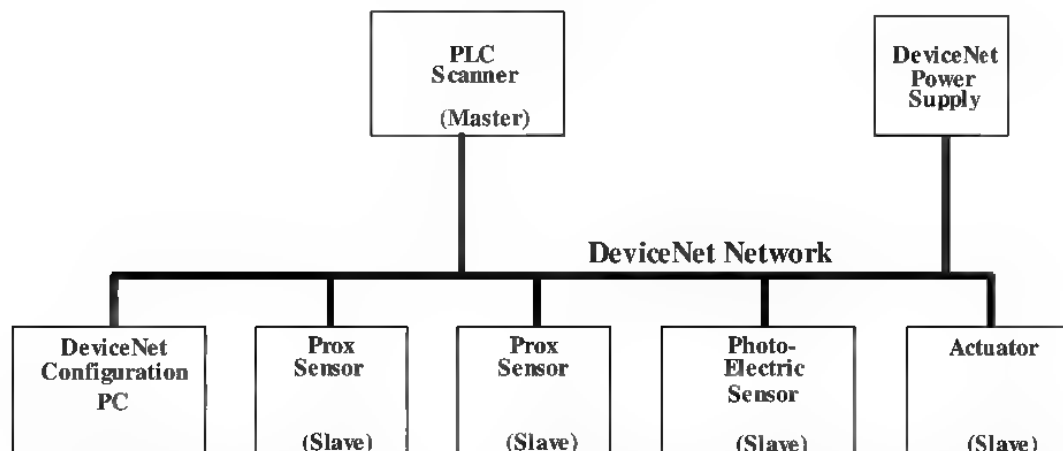
The preceding chapters presented the “*general model*” rules for establishing connections between devices. The general model calls for the utilization of an Explicit Messaging Connection to manually create and configure Connection Objects within each connection end-point. This chapter uses the general model as a basis for the definition of a set of connections, which facilitate communications typically seen in a Master/Slave relationship. These Connections are referred to collectively as the *Predefined Master/Slave Connection Set*.

The **Master** is the device that gathers and distributes I/O data for the process controller. **Slaves** are the devices from which the Master gathers I/O data and to which the Master distributes I/O data.

The Master “owns” the Slaves who’s MAC IDs appear in its scan list. To determine with what Slaves it will communicate, the Master examines its scan list and sends commands accordingly. Except for the Duplicate MAC ID Check, a Slave cannot initiate any communication before being told by the Master to do so.

Figure 3-6.1 illustrates a single Master with multiple Slaves.

Figure 3-6.1 Example DeviceNet Master/Slave Implementation



Many of the steps involved in the creation and configuration of an Application to Application connection have been removed within the Predefined Master/Slave Connection Set definition. This, in turn, presents the means by which a communication environment can be established using less network and device resources.

The following terms are used:

- **Group 2 Server:** A UCMM capable device that has been told to act as the Server for the Predefined Master/Slave Identifier Connections. See *DeviceNet Slave*.
- **Group 2 Client:** A device that has gained ownership of the Predefined Master/Slave Connection Set within a Server such that it can act as the Client on those connections. See *DeviceNet Master*.
- **UCMM Capable Device:** A device that supports the Unconnected Message Manager (UCMM). At minimum, this requires support for reception and processing of Unconnected Request Messages described in Chapter 2, section 2-7.2, UCMM Services.

- **UCMM Incapable Device:** Typically a low-level device that, because of network interrupt management and first generation CAN chip screening capabilities, does not support the UCMM.
- **Group 2 Only Server:** A slave (server) device that is UCMM incapable and must use the Predefined Master/Slave Connection Set to establish communications (at a minimum, the Predefined Master/Slave Explicit Messaging Connection must be supported). A Group 2 Only device can transmit and receive only those identifiers defined by the Predefined Master/Slave Connection Set. (See Figure 3-7.1 Predefined Master/Slave Connection Set Identifier Fields.)
- **Group 2 Only Client:** A device that is acting as a Group 2 Client to a Group 2 Only Server. The Group 2 Only Client provides the UCMM functionality described in Chapter 2, section 2-7.2, UCMM Services, for Group 2 Only Servers that it has allocated. This concept is described in more detail later in this chapter.
- **DeviceNet Master:** Refers to a type of application called Master/Slave. The DeviceNet Master is the device that gathers and distributes I/O data for the process controller. A Master scans its Slave devices based on a scan list it contains. With respect to the network, the Master is a Group 2 Client or a Group 2 Only Client.
- **DeviceNet Slave:** Refers to a type of application called Master/Slave. A Slave returns I/O data to its Master when it is scanned. With respect to the network, the Slave is a Group 2 Server or a Group 2 Only Server.
- **Predefined Master/Slave Connection Set:** A set of Connections that facilitate communications typically seen in a Master/Slave relationship. Many of the steps involved in the creation and configuration of an Application to Application connection have been removed within the Predefined Master/Slave Connection Set definition. This, in turn, presents the means by which a communication environment can be established using less network and device resources.

### 3-7 Predefined Master/Slave Connection Set Messages

The CAN Identifier Fields associated with the Predefined Master/Slave Connection Set are shown in Figure 3-7.1. Figure 3-7.1 defines the Identifiers that are to be used with all connection based messaging involved in the Predefined Master/Slave Connection Set and, as such, it also illustrates **produced\_connection\_id** and **consumed\_connection\_id** attributes associated with Predefined Master/Slave Connection Objects.

Figure 3-7.1 Predefined Master/Slave Connection Set Identifier Fields

IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MAC ID						Group 1 Messages	000 – 3ff
0	1	1	0	0	Source MAC ID						Slave's I/O Multicast Poll Response Message	
0	1	1	0	1	Source MAC ID						Slave's I/O Change of State or Cyclic Message	
0	1	1	1	0	Source MAC ID						Slave's I/O Bit–Strobe Response Message	
0	1	1	1	1	Source MAC ID						Slave's I/O Poll Response or Change of State/Cyclic Acknowledge Message	
1	0	MAC ID					Group 2 Message ID				Group 2 Messages	400 – 5ff
1	0	Source MAC ID					0	0	0		Master's I/O Bit Strobe Command Message	
1	0	Multicast MAC ID					0	0	1		Master's I/O Multicast Poll Command Message	
1	0	Destination MAC ID					0	1	0		Master's Change of State or Cyclic Acknowledge Message	
1	0	Source MAC ID					0	1	1		Slave's Explicit/ Unconnected Response Messages	
1	0	Destination MAC ID					1	0	0		Master's Explicit Request Messages	
1	0	Destination MAC ID					1	0	1		Master's I/O Poll Command/Change of State/Cyclic Message	
1	0	Destination MAC ID					1	1	0		Group 2 Only Unconnected Explicit Request Messages (reserved)	
1	0	Destination MAC ID					1	1	1		Duplicate MAC ID Check Messages	

**Important:** Group 2, Message ID = 6 is reserved for use as the *Group 2 Only Unconnected Explicit Request* message port and should not be used for any other purpose.

The following types of messages are mentioned in Figure 3 7.1.

- **I/O Bit–Strobe Command/Response Messages:** The Bit Strobe Command is an I/O Message that is transmitted by the Master. A Bit–Strobe Command Message has multi–cast capabilities. Multiple Slaves can receive and react to the same Bit–Strobe Command (multi–cast capabilities). The Bit–Strobe Response is an I/O Message that a Slave transmits back to the Master when the Bit–Strobe Command is received. Within a Slave, the Bit–Strobe Command and Response Messages are received/transmitted by a single Connection Object.
- **I/O Poll Command/Response Messages:** The Poll Command is an I/O Message that is transmitted by the Master. A Poll Command is directed towards a single, specific Slave (point–to–point). A Master must transmit a separate Poll Command Message for each one of its Slaves that is to be polled. The Poll Response is an I/O Message that a Slave transmits back to the Master when the Poll Command is received. Within a Slave, the Poll Command and Response Messages are received/transmitted by a single Connection Object.

- ***I/O Change of State/Cyclic Messages:*** The Change of State/Cyclic Message is transmitted by either the Master or the Slave. A Change of State/Cyclic Message is directed towards a single specific node (point-to-point). An Acknowledge Message may be returned in response to this message. Within either the Master or the Slave, the producing Change of State Message and consuming Acknowledge Message are received/transmitted by one connection object. The consuming Change of State Message and producing Acknowledge Message are received/transmitted by a second connection object.
- ***I/O Multicast Poll Messages:*** The Multicast Poll *Command* is an I/O Message that is transmitted by the Master. A Multicast Poll is directed towards one or more Slaves. The Multicast Poll *Response* is an I/O Message that a Slave transmits back to the Master when the Multicast Poll Command is received. Within a Slave, the Multicast Poll Command and Response Messages are received/transmitted by a single Connection Object.
- ***Explicit Response/Request Messages:*** Explicit Request Messages are used to perform operations such as reading and writing attributes. Explicit Response Messages indicate the results of the attempt to service an Explicit Request Message. Within a Slave, Explicit Requests and Responses are received/transmitted by a single Connection Object.
- ***Group 2 Only Unconnected Explicit Request Messages:*** The *Group 2 Only Unconnected Explicit Request* port is used to allocate/release the Predefined Master/Slave Connection Set. This port (Group 2, Message ID = 6) is reserved and should not be used for any other purpose.
- ***Group 2 Only Unconnected Explicit Response Messages:*** The *Group 2 Only Unconnected Explicit Response* port is used to respond to *Group 2 Only Unconnected Explicit Request* messages and to send Device Heartbeat / Device Shutdown messages. These messages are transmitted using the same identifier (Group 2, Message ID = 3) as Explicit Response messages.
- **Duplicate MAC ID Check Message:** This is defined in Chapter 2, section 2-3.2.

Note that all but two of the messages associated with the Predefined Master/Slave Connection Set are transmitted across Message Group 2.

**Important:** Other devices, in addition to Group 2 Only Servers, can establish Connections that utilize Message Group 2. Group 2 Servers and Group 2 Only Servers may exist on a network where other devices are making use of Message Group 2. Be aware that when this scenario exists, Group 2 interrupts to devices that are Group 2 Only may increase.



## 3-8 Slave Connection Object Characteristics

This section presents the externally visible characteristics of the Connection Objects associated with the Predefined Master/Slave Connection Set **within Slave devices**. The Predefined Master/Slave Connection Objects defined for Slave devices are:

- The Bit-Strobe Connection - Responsible for receiving the Master's Bit-Strobe Command and returning the associated Bit-Strobe Response.
- The Poll Connection - Responsible for receiving the Master's Poll Command and returning the associated Poll Response.
- The Explicit Messaging Connection - Responsible for the reception of Explicit Requests and returning associated responses.
- The Change of State/Cyclic Connection - Responsible for the sending the Change of State/Cyclic Message and possibly receiving the Acknowledge Response.
- The Multicast Poll Connection - Responsible for receiving the Master's Multicast Poll Command and returning the associated Multicast Poll Response.

**Important:** This section further refines information presented in Volume 1, Chapter 3 for use by Connection Objects within the Predefined Master/Slave Connection Set. Except where noted, all information specified in Volume 1, Chapter 3 (e.g. Services, Attributes, etc.) applies to the Connection Objects described in this section.

### 3-8.1 Connection Instance IDs

Every Connection Object in existence has an assigned **Connection Instance ID**, which identifies the Connection Object amongst several Connection Objects within the Connection Class. The Instance IDs that must be utilized by a Slave device to identify Predefined Master/Slave Connection Objects are shown in Table 3-8.1.

**Table 3-8.1 Connection Instance ID for Predefined Master/Slave Connections**

Connection Instance ID #	Description
1	References the Explicit Messaging Connection into the Server
2	References the Poll I/O Connection
3	References the Bit-Strobe I/O Connection
4	References the Slave's Change of State or Cyclic I/O Connection
5	References the Multicast Poll I/O Connection

**Important:** A Slave must reserve the Instance IDs from Table 3-8.1 for the Predefined Master/Slave Connections that it supports. For example, if a device supports the Polled I/O Connection, it must reserve/utilize Connection Instance ID #2 to identify the Polled I/O Connection Object. If a device does not support the Poll Connection, then it is free to allocate Connection Instance ID #2 to identify some other Connection Object.

### 3-8.2 Slave Connection Instance Attributes

This section defines default attribute values utilized by a Slave device within Predefined Master/Slave Connection Objects.

Table 3-8.2 through Table 3-8.7 defines attribute values for the Predefined Master/Slave I/O Connection Objects. These default values are initially loaded into the attributes when the Connection Object transitions from the **Non-existent** state to the **Configuring** state (see section 3-8.3)

**Table 3-8.2 Default Change of State/Cyclic Connection Object Attribute Values (Acknowledged)**

Attribute ID (decimal)	Attribute Name	Default Value	Description
1	state	01	Indicates this object is in the <b>Configuring</b> state.
2	instance_type	01	Indicates this is an I/O Connection
3	transportClass trigger	02 <sub>hex</sub> or 03 <sub>hex</sub> OR 12 <sub>hex</sub> or 13 <sub>hex</sub>	For a cyclic connection use 02h or 03h. For a change of state connection use 12h or 13h. (Class 2 or 3 clients have the same behavior.)
4	produced_connection_id		See Figure 3-7.1 - Slave's I/O Change of State or Cyclic Message
5	consumed_connection_id		See Figure 3-7.1 - Master's Change of State or Cyclic Acknowledge Message
6	initial_comm_characteristics	01	Indicates that the Slave's Change of State or Cyclic Connection produces across Message Group 1 and consumes across Message Group 2. This value also indicates that the Slave's MAC ID appears in the CAN Identifier Field of the Group 2 Message that the Slave will consume.
7	produced_connection_size		No specified default. An implementation must initialize this attribute according to the Application Object that is referenced by the default <b>produced_connection_path</b> attribute
8	consumed_connection_size	0	Connection size defaults to zero length acknowledge.
9	expected_packet_rate	0	Expected packet rate must be configured.
12	watchdog_timeout_action	0	Transition to the <b>Timed Out</b> state.
13	produced_connect.on_path_length		No specified default. An implementation must initialize this attribute with the number of bytes in the default <b>produced_connection_path</b> attribute
14	produced_connection_path		No specified default. An implementation <b>must</b> choose an Application Object to reference by default and initialize this attribute accordingly.
15	consumed_connection_path_length	4	Path length for Acknowledge Handler Object.
16	consumed_connection_path	20h 2Bh 24h 01h	Path set to instance 1 of the Acknowledge Handler Object.
17	production_inhibit_time	0	Default is no inhibit time.

Table 3-8.3 Default Change of State/Cyclic Connection Object Attribute Values (Unacknowledged)

Attribute ID (decimal)	Attribute Name	Default Value	Description
1	state	01	Indicates this object is in the <b>Configuring</b> state.
2	instance_type	01	Indicates this is an I/O Connection.
3	transportClass_trigger	00 <sub>hex</sub> OR 10 <sub>hex</sub>	For a cyclic connection use 00h . For a change of state connection use 10h. (Class 2 or 3 clients have the same behavior )
4	produced_connection_id		Slave's I/O Change of State or Cyclic Message
5	consumed_connection_id	0FFFF <sub>hex</sub>	Value assigned when instance is not consuming data.
6	initial_comm_characteristics	0F <sub>hex</sub>	Indicates that the Slave's Change of State or Cyclic Connection produces across Message Group 1 and does not consume.
7	produced_connection_size		No specified default. An implementation must initialize this attribute according to the Application Object that is referenced by the default <b>produced_connection_path</b> attribute.
8	consumed_connection_size	0	Connection consumes no data.
9	expected_packet_rate	0	Expected packet rate must be configured.
12	watchdog_timeout_action	0	Transition to the <b>Timed Out</b> state
13	produced_connection_path_length		No specified default. An implementation must initialize this attribute with the number of bytes in the default <b>produced_connection_path</b> attribute.
14	produced_connection_path		No specified default. An implementation <b>must</b> choose an Application Object to reference by default and initialize this attribute accordingly.
15	consumed_connection_path_length	0	Connection consumes no data.
16	consumed_connection_path	Empty	Connection consumes no data.
17	production_inhibit_time	0	Default is no inhibit time.

Table 3-8.4 Default Poll Connection Object Attribute Values

Attribute ID (decimal)	Attribute Name	Default Value	Description
1	state	01	Indicates the Poll Connection Object is in the <b>Configuring</b> state
2	instance_type	01	Indicates this is an I/O Connection.
3	transportClass_trigger	82 hex OR 83 hex	Server/Transport Class 2 OR Server/Transport Class 3 Implementations can choose either Transport Class 2 or 3 as the default.
4	produced_connection_id		See Figure 3-7.1 - Slave's I/O Poll Response Message
5	consumed_connection_id		See Figure 3-7.1 - Master's I/O Poll Command Message
6	initial_comm_characteristics	01	Indicates that the Slave's Poll I/O Connection produces across Message Group 1 and consumes across Message Group 2. This value also indicates that the Slave's MAC ID appears in the CAN Identifier Field of the Group 2 Message that the Slave will consume. This information is reflected in Figure 3-7.1.
7	produced_connection_size		No specified default. An implementation must initialize this attribute according to the Application Object that is referenced by the default <b>produced_connection_path</b> attribute.
8	consumed_connection_size	0	No specified default. An implementation must initialize this attribute according to the Application Object that is referenced by the default <b>consumed_connection_path</b> attribute.
9	expected_packet_rate	0	Expected packet rate must be configured
12	watchdog_timeout_action	0	Transition to the <b>Timed Out</b> state.
13	produced_connection_path_length		No specified default. An implementation must initialize this attribute with the number of bytes in the default <b>produced_connection_path</b> attribute.
14	produced_connection_path		No specified default. An implementation <b>must</b> choose an Application Object to reference by default and initialize this attribute accordingly.
15	consumed_connection_path_length		No specified default. An implementation must initialize this attribute with the number of bytes in the default <b>consumed_connection_path</b> attribute.
16	consumed_connection_path		No specified default. An implementation <b>must</b> choose an Application Object to reference by default and initialize this attribute accordingly.
17	production_inhibit_time	0	Default is no inhibit time

Table 3-8.5 Default Bit-Strobe Connection Object Attribute Values

Attribute ID (decimal)	Attribute Name	Default Value	Description
1	State	01	Indicates the Bit-Strobe Connection Object is in the <b>Configuring</b> state
2	instance_type	01	Indicates this is an I/O Connection.
3	transportClass_trigger	82 <sub>hex</sub> OR 83 <sub>hex</sub>	Server/Transport Class 2 OR Server/Transport Class 3. Implementations can choose either Transport Class 2 or 3 as the default.
4	produced_connection_id		See Figure 3-7.1 - Slave's I/O Bit-Strobe Response Message
5	consumed_connection_id		See Figure 3-7.1 - Master's I/O Bit-Strobe Command Message
6	initial_comm_characteristics	02	Indicates that the Slave's Bit-Strobe I/O Connection produces across Message Group 1 and consumes across Message Group 2. This value also indicates that the Master's MAC ID appears in the CAN Identifier Field of the Group 2 Message that the Slave will consume. This information is reflected in Figure 3-7.1.
7	produced_connection_size		No specified default. An implementation must initialize this attribute according to the Application Object that is referenced by the default <b>produced_connection_path</b> attribute. <u><b>Note that section 3-10.2 indicates that a Bit-Strobe Response can be no longer than 8 bytes in length. With respect to the Bit-Strobe Connection, this attribute can never contain a value greater than 8 in the Established state.</b></u>
8	consumed_connection_size	8	This attribute shall be set to a value of 8 (see section 3-10.1).
9	expected_packet_rate	0	Expected packet rate must be configured
12	watchdog_timeout_action	0	Transition to the <b>Timed Out</b> state.
13	produced_connection_path_length		No specified default. An implementation must initialize this attribute with the number of bytes in the default <b>produced_connection_path</b> attribute.
14	produced_connection_path		No specified default. An implementation <b>must</b> choose an Application Object to reference by default and initialize this attribute accordingly.
15	consumed_connection_path_length		No specified default. An implementation must initialize this attribute with the number of bytes in the default <b>consumed_connection_path</b> attribute.
16	consumed_connection_path		No specified default. An implementation <b>must</b> choose an Application Object to reference by default and initialize this attribute accordingly.
17	production_inhibit_time	0	Default is no inhibit time

Table 3-8.6 Default Multicast Poll Connection Object Attribute Values

Attribute ID (decimal)	Attribute Name	Default Value	Description
1	state	01	Indicates the Multicast Poll Connection Object is in the <b>Configuring</b> state
2	Instance type	01	Indicates this is an I/O Connection.
3	transportClass_trigger	82 hex OR 83 hex	Server/Transport Class 2 OR Server/Transport Class 3 Implementations can choose either Transport Class 2 or 3 as the default.
4	produced_connection_id		See Figure 3-7.1 - Slave's I/O Multicast Poll Response Message
5	consumed_connection_id	0xFFFF	The slave receives the consumed connection ID from the master after allocation.
6	initial_comm_characteristics	01	Indicates that the Slave's Multicast Poll I/O Connection produces across Message Group 1 and consumes across Message Group 2. This information is reflected in Figure 3-7.1.
7	produced_connection_size		No specified default. An implementation must initialize this attribute according to the Application Object that is referenced by the default <b>produced_connection_path</b> attribute.
8	consumed_connection_size		No specified default. An implementation must initialize this attribute according to the Application Object that is referenced by the default <b>consumed_connection_path</b> attribute.
9	expected_packet_rate	0	Expected packet rate must be configured
12	watchdog_timeout_action	0	Transition to the <b>Timed Out</b> state.
13	produced_connection_path_length		No specified default. An implementation must initialize this attribute with the number of bytes in the default <b>produced_connection_path</b> attribute.
14	produced_connection_path		No specified default. An implementation <b>must</b> choose an Application Object to reference by default and initialize this attribute accordingly.
15	consumed_connection_path_length		No specified default. An implementation must initialize this attribute with the number of bytes in the default <b>consumed_connection_path</b> attribute.
16	consumed_connection_path		No specified default. An implementation <b>must</b> choose an Application Object to reference by default and initialize this attribute accordingly.
17	production_inhibit_time	0	Default is no inhibit time

defines attribute values for the Predefined Master/Slave Explicit Messaging Connection Object. These default values are initially loaded into the attributes when the Connection Object transitions from the **Non-existent** state to the **Established** state (see Chapter 5, section 5-3.4.2).

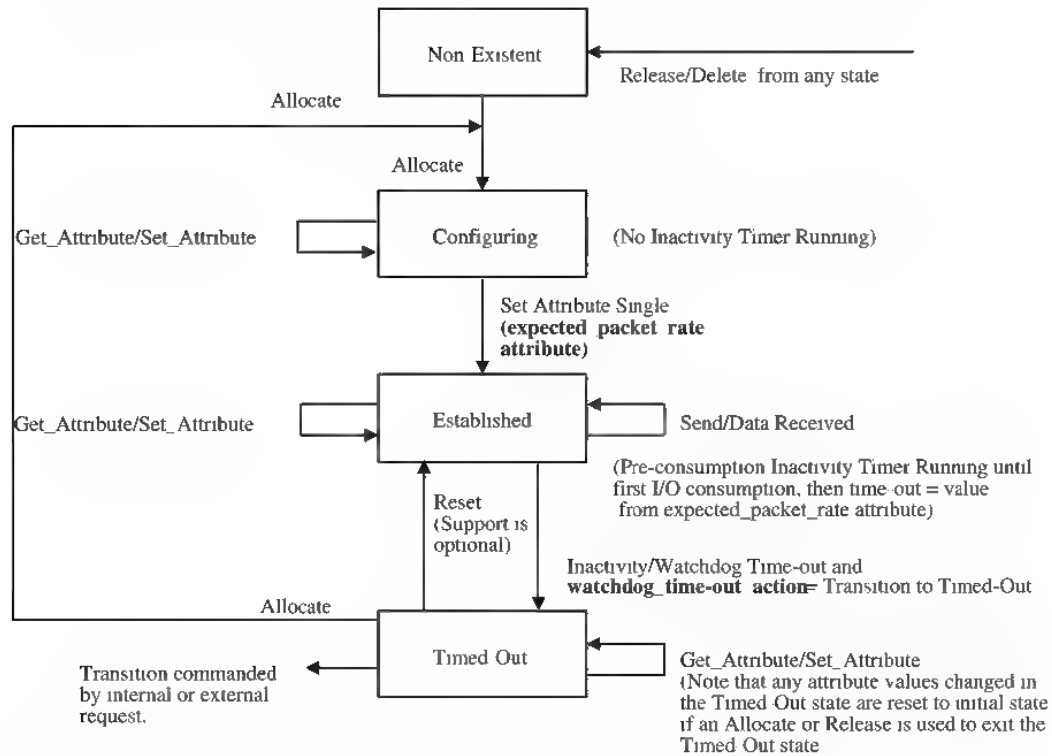
**Table 3-8.7 Predefined Master/Slave Explicit Messaging Connection Object Attribute Values**

Attribute ID (decimal)	Attribute Name	Default Value	Description
1	state	03	Indicates the Explicit Messaging is in the <b>Established</b> state.
2	instance_type	00	Indicates this is an Explicit Messaging Connection.
3	transportClass_trigger	83 <sub>hex</sub>	Server/Transport Class 3.
4	produced_connection_id		See Figure 3-7.1 - Slaves Explicit Response Messages.
5	consumed_connection_id		See Figure 3-7.1 - Masters Explicit Request Messages
6	initial_comm_characteristics	21 <sub>hex</sub>	Indicates that the Slave's Explicit Messaging Connection produces and consumes across Message Group 2. Additionally, this value indicates that the Slave's MAC ID appears in the CAN Identifier Fields of the Group 2 Messages that the Slave consumes and produces. This information is reflected in Figure 3-7.1.
7	produced_connection_size		No specified default. This must be initialized as described in Volume 1, Chapter 3-4.3.
8	consumed_connection_size		No specified default. This must be initialized as described in Volume 1, Chapter 3-4.3.
9	expected_packet_rate	09C4 <sub>hex</sub>	2500 milliseconds as described in Volume 1, Chapter 3-4.3.
12	watchdog_timeout_action	1	<b>Auto_Delete</b> as described in Volume 1, Chapter 3-4.3. This attribute shall be settable to Deferred Delete as described in Volume 1, Chapter 3-4.3.
13	produced_connection_path_length	0	As described in Volume 1, Chapter 3-4.3.
14	produced_connection_path	Empty	As described in Volume 1, Chapter 3-4.3.
15	consumed_connection_path_length	0	As described in Volume 1, Chapter 3-4.3.
16	consumed_connection_path	Empty	As described in Volume 1, Chapter 3-4.3.
17	production_inhibit_time	0	Default is no inhibit time

### 3-8.3 Predefined Master/Slave Connection Instance Behavior

Figure 3-8.1 illustrates the Predefined Master/Slave Connection Set I/O Connection Object State Transition Diagram.

Figure 3-8.1 Predefined Master/Slave I/O Connection State Transition Diagram



Note that the Allocate and Release services send the connection instance back to initial state. All Connection Object attributes are reset to their default values.

**Important:** As far as attribute modification is concerned, Predefined Master/Slave I/O Connections must (at a minimum) support the modification of the **expected\_packet\_rate** attribute.

The SEM presented below provides a formal definition of the behavior of I/O Connections within the Predefined Master/Slave Connection Set. This SEM inherits from and/or overrides actions presented in Table 3-4.20 I/O Connection State Event Matrix in Volume 1, Chapter 3-4.7.1. Actions that override behavior are highlighted. Pay close attention to overridden actions.

**Important:** The State Event Matrix presented below does not dictate rules with regards to product specific, internal logic. Any attempt to access the Connection Class or a Connection Object Instance may need to pass through product specific verification. This may result in an error scenario that is not indicated by the SEM Table 3-8.8. This may also result in additional, product specific indications delivered from a Connection Object to the application and/or a specific Application Object. The point to remember is that the Predefined Master/Slave I/O Connection Object must exhibit the externally visible behavior specified by the SEM below and the attribute definitions (section 3-8.4).



Table 3-8.8 Predefined Master/Slave I/O Connection State Event Matrix

Event	I/O Connection Object State			
	Non-Existent	Configuring	Established	Timed Out
DeviceNet Object receives an <i>Allocate Master/Slave Connection Set</i> Request that passes all error checks specified in Volume 1, Chapter 3-5.2. This request specifies one of the Predefined Master/Slave I/O Connections.	Instantiate a Connection Object for each requested I/O Connection & set attributes to default values specified in section 3-8.2. Transition to Configuring	This is an error scenario described in Chapter 3, section 3-5 2.	This is an error scenario described in Chapter 3, section 3-5 2.	Set attributes to default values specified in section 3-8 2. Transition to Configuring
Connection Class receives a Delete Request or the UCMM receives a Close request and the request specifies a Predefined Master/Slave I/O Connection Object.	Same as described in Volume 1, Chapter 3, Table 3-4.20	Release all associated resources. Transition to Non-existent. <sup>1</sup>	Release all associated resources. Transition to Non-existent. <sup>1</sup>	Release all associated resources. Transition to Non-existent. <sup>1</sup>
DeviceNet Object receives a <i>Release Master/Slave Connection Set</i> Request that passes all error checks specified in Volume 1, Chapter 3-5.2. This request specifies one of the Predefined Master/Slave I/O Connections.	This is an error scenario described in Volume 1, Chapter 3-4.3.	Release all associated resources. Transition to Non-existent. <sup>1</sup>	Release all associated resources. Transition to Non-existent. <sup>1</sup>	Release all associated resources. Transition to Non-existent. <sup>1</sup>
Set Attribute Single	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Validate/service the request based on internal logic and per the Access Rules presented in section 3-8 4. If this is a valid request to set the <code>expected_packet_rate</code> attribute, then perform the steps specified in Volume 1, Chapter 3, Table 3-4.20 under the <code>Apply_Attributes</code> Event/Configuring State and transition to Established. Return appropriate response.	Validate/service the request based on internal logic and per the Access Rules presented in section 3-8 4. Return appropriate response.	Validate/service the request based on internal logic and per the Access Rules presented in section 3-8 4. Return appropriate response.
Get_Attribute_Single	Same as described in Volume 1, Chapter 3, Table 3-4 20	Validate/service the request based on internal logic and per the Access Rules presented in section 3-8.4. Return appropriate response.	Validate/service the request based on internal logic and per the Access Rules presented in section 3-8.4. Return appropriate response.	Validate/service the request based on internal logic and per the Access Rules presented in section 3-8.4. Return appropriate response
Reset	Same as described in Volume 1, Chapter 3, Table 3-4 20.	Same as described in Volume 1, Chapter 3, Table 3-4 20.	Same as described in Volume 1, Chapter 3, Table 3-4 20.	Same as described in Volume 1, Chapter 3, Table 3-4 20.

Event	I/O Connection Object State			
	Non-Existent	Configuring	Established	Timed Out
Apply_Attributes	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20. <sup>2</sup>	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20.
Receive_Data	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20.
Send_Message	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20.
Inactivity/Watchdog Timer expires	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20.	Same as described in Volume 1, Chapter 3, Table 3-4.20. <sup>1</sup>	Same as described in Volume 1, Chapter 3, Table 3-4.20.

<sup>1</sup> Whenever a Connection Object within the Predefined Master/Slave Connection Set transitions out of the Established state, the entire Predefined Master/Slave Connection Set may need to be automatically released. When the Predefined Master/Slave Connection Set is released, all associated Connection Objects return to the Non-existent state. See section 3-5.2 for more details concerning the automatic release of the Predefined Master/Slave Connection Set.

<sup>2</sup> Since an implied Apply\_Attributes accompanies a Set\_Attribute\_Single of the expected\_packet\_rate attribute of a Predefined Master/Slave I/O Connection Object in the Configuring state, the only time an Apply\_Attributes Explicit Messaging Request will succeed is when the expected\_packet\_rate has yet to be successfully modified via the Set\_Attribute\_Single request and, thus, still contains the default value of zero (0).

If an implementation detects that it does not support an Explicit Messaging Service indicated in Table 3-8.8, then an Error Response specifying *Service Not Supported* (General Error Code 08) is returned.

The Explicit Messaging Connection associated with the Master/Slave Connection set behaves the same as any other Explicit Messaging Connection. The means by which it comes into and goes out of existence are its distinguishing characteristic (e.g. allocation via a request sent to the DeviceNet Object vs. creation via the UCMM). The State Transition Diagram below highlights the behavior of the Predefined Master/Slave Explicit Messaging Connection. The **watchdog\_timeout** action attribute value of *Deferred Delete* shall be supported.

Figure 3-8.2 Predefined Master/Slave Explicit Messaging State Transition Diagram

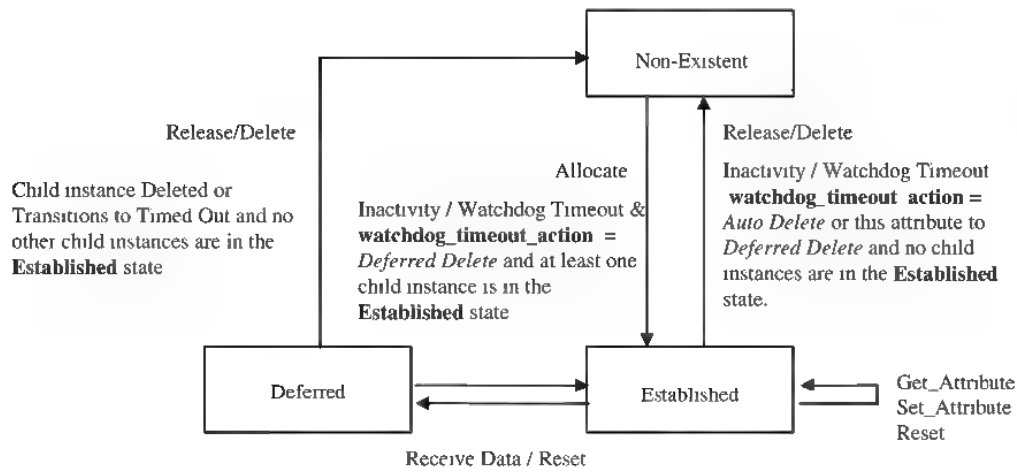


Table 3-8.9 provides a detailed State Event Matrix for the Predefined Master/Slave Explicit Messaging Connection Object. The SEM presented below provides a formal definition of the behavior of the Explicit Messaging Connection within the Predefined Master/Slave Connection Set. This SEM inherits from and/or overrides actions presented in Table 3-4.22, Explicit Messaging Connection State Event Matrix in Volume 1, Chapter 3-4.6.3.

Table 3-8.9 Predefined Master/Slave Explicit Messaging Connection State Event Matrix

Event	Explicit Messaging Connection Object State		
	Non-Existent	Established	Deferred
DeviceNet Object receives an <i>Allocate Master/Slave Connection Set Request</i> that passes all error checks specified in Volume 1, section 3-4.3. This request specifies the Predefined Master/Slave Explicit Messaging Connection.	Instantiate the Predefined Master/Slave Explicit Messaging Connection Object for each requested I/O Connection & set attributes to default values (see section 3-8.2). Transition to Established.	This is an error scenario described in section 3-5.2	This is an error scenario described in section 3-5.2
UCMM receives a Close Request or the Connection Class receives a Delete Request and the request specifies the Predefined Master/Slave Explicit Messaging Connection Object.	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22. <sup>1</sup>	Same as Volume 1, Chapter 3, Table 3-4.22. <sup>1</sup>
DeviceNet Object receives a <i>Release Master/Slave Connection Set Request</i> that passes all error checks specified in Volume 1, section 3-4.3. This request specifies the Predefined Master/Slave Explicit Messaging Connection.	This is an error scenario described in section 3-5.2.	Release all associated resources. Transition to Non-existent. <sup>1</sup>	Release all associated resources. Transition to Non-existent. <sup>1</sup>
Set_Attribute_Single	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.
Get_Attribute_Single	Same as Volume 1, Chapter 3, Table 3-4.22	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22
Reset	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.

Apply_Attributes	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.
Receive_Data	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.
Send_Message	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22.
Inactivity/Watchdog Timer expires	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22. <sup>1</sup>	Same as Volume 1, Chapter 3, Table 3-4.22. <sup>1</sup>
Child connection instance Deleted or Transitions to Timed Out	Same as Volume 1, Chapter 3, Table 3-4.22.	Same as Volume 1, Chapter 3, Table 3-4.22. <sup>1</sup>	Same as Volume 1, Chapter 3, Table 3-4.22. <sup>1</sup>

<sup>1</sup> Whenever a Connection Object within the Predefined Master/Slave Connection Set transitions out of the Established state, the entire Predefined Master/Slave Connection Set may need to be automatically released. When the Predefined Master/Slave Connection Set is released, all associated Connection Objects return to the Non-existent state. See section 3-5.2 for more details concerning the automatic release of the Predefined Master/Slave Connection Set.

If an implementation detects that it does not support an Explicit Messaging Service indicated in Table 3-8.9, then an Error Response specifying *Service Not Supported* (General Error Code 08) is returned.

### 3-8.4 Connection Instance Attribute Access Rules

Refer to Volume 1, Chapter 3-4.7 for a general description of the Access Rules associated with Connection Objects. Table 3-8.10 presents Access Rules specific to the Predefined Master/Slave I/O Connection Objects. Table 3-8.10 inherits from and/or overrides rules presented in Volume 1, Chapter 3-4.7. Rules that override behavior are highlighted. Pay close attention to overridden rules.

**Table 3-8.10 Predefined Master/Slave I/O Connection Object Attribute Access**

Attribute	I/O Connection State			
	Non-Existent	Configuring	Established	Timed Out
State	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23
instance_type	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23
transport Class trigger	See Vol 1, Chpt 3, Table 3-4.23	Get/Set <sup>1</sup>	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23
produced_connection_id	See Vol 1, Chpt 3, Table 3-4.23	Get Only	Get Only	Get Only
consumed_connection_id	See Vol 1, Chpt 3, Table 3-4.23	Get Only <sup>3</sup>	Get Only <sup>3</sup>	Get Only <sup>3</sup>
initial_comm_characteristics	See Vol 1, Chpt 3, Table 3-4.23	Get Only	Get Only	Get Only
produced_connection_size	See Vol 1, Chpt 3, Table 3-4.23	Get/Set <sup>2</sup>	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23
consumed_connection_size	See Vol 1, Chpt 3, Table 3-4.23	Get Only	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23
expected_packet_rate	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23
watchdog_timeout_action	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23

produced_connection_path_length	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23
produced_connection_path	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23
consumed_connection_path_length	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23
consumed_connection_path	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23	See Vol 1, Chpt 3, Table 3-4.23

<sup>1</sup> The transportClass\_trigger attribute of a Predefined Master/Slave I/O Connection Object within a Slave device can only be modified to one the following values:

82<sub>hex</sub> - Server/Transport Class 2

83<sub>hex</sub> - Server/Transport Class 3

<sup>2</sup> The produced\_connection\_size attribute of the Bit-Strobe I/O Connection cannot be set to a value greater than 8.

<sup>3</sup> This attribute shall have Get/Set access within the Multicast Poll Connection

The Access Rules for the Predefined Master/Slave Explicit Messaging Connection are identical to the Explicit Messaging Connection Access Rules presented in Volume 1, Chapter 3-4.7.3.

### 3-9 Master Connection Object Characteristics

This specification does not present a large number of characteristics with respect to Connection Objects within a Master. It is assumed that the Master understands how it has configured its Slaves and will exhibit the external behavior necessary to properly interface with them.

### 3-10 Bit-Strobe Command/Response Messages

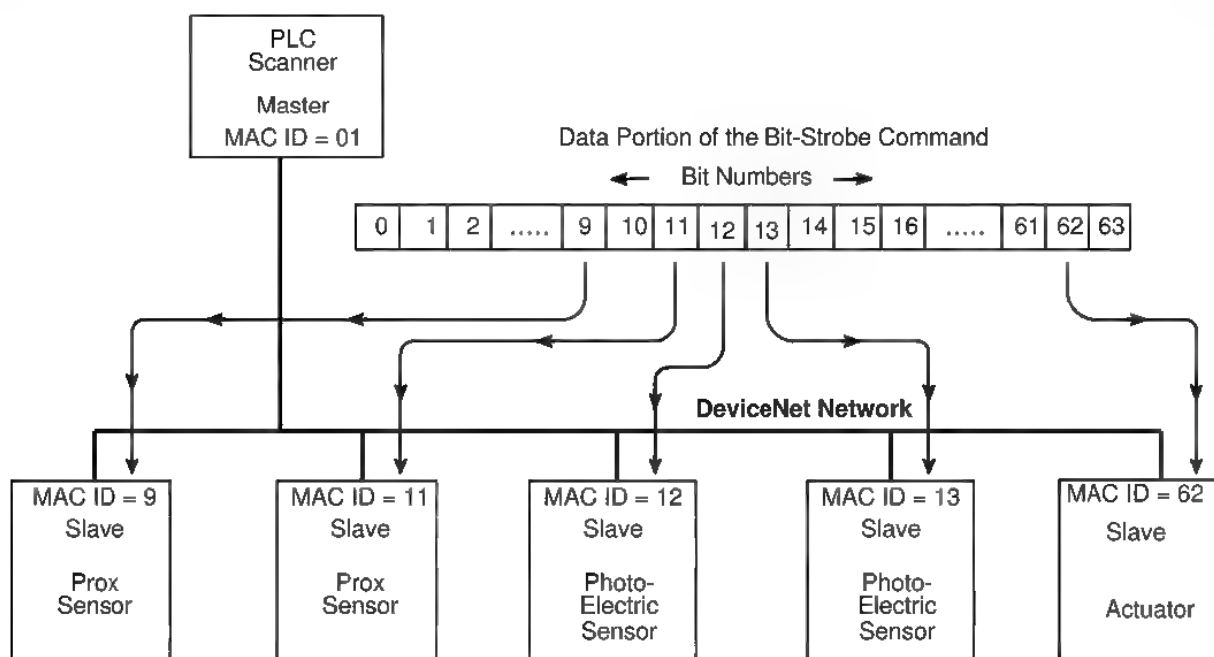
Bit-Strobe Command and Response messages rapidly move *small amounts* of I/O data between a Master and its *Bit-Strobed* Slaves.

#### 3-10.1 Bit-Strobe Command Message

The ***Bit-Strobe Command***, sends one bit of output data to each Slave whose MAC ID appears in the Master's scan list.

The Bit-Strobe Command message contains a bit string of 64 bits (8 bytes) of output data, one output bit per MAC ID on the network. One bit is assigned to each MAC ID supported on the network (0...63) as shown in Figure 3-10.1.

Figure 3-10.1 DeviceNet Bit-Strobe Command



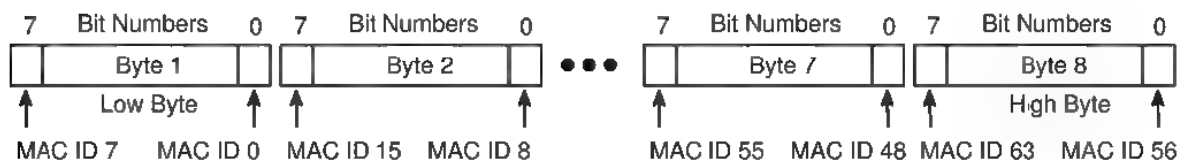
Note: The MAC IDs and bit number are shown in decimal.

In Figure 3-10.1 the Master is sending one bit to each of its five Slaves. Note that the Slave's MAC IDs are not required to be sequential. The entire 64-bit string is transmitted, regardless of the number of Slave devices and their MAC IDs. Only those slaves configured to consume the Bit Strobe Command do so.

The format of the Bit-Strobe Command's 64-bit string is (see Figure 3-10.2):

- The low-order bit in the lowest-order byte is assigned to MAC ID 0.
- The high-order bit in the highest-order byte is assigned to MAC ID 63.

Figure 3-10.2 Bit-Strobe Bit versus Byte Ordering: Wire View



A Slave device can be designed to do one or all of the following:

- Ignore the Bit-Strobe Command (perhaps the Slave is a Polled device, or the Bit-Strobe is not Allocated)
- Consume the Bit-Strobe Command and its output data
- Consume the Bit-Strobe Command as a trigger and ignore the output data

Slaves default to ignoring the Bit-Strobe Command until the Bit-Strobe Connection is allocated. See section 3-5, for information about allocation.

**Important:** Not all DeviceNet devices are required to use the output bit assigned to them. For example, an input device can ignore the output bit, but use the Bit–Strobe Command to trigger the transmission of return input data in a Bit–Strobe Response message.

**Important:** A Bit–Strobe Command message transmitted with no Application I/O data in the CAN Data Field is interpreted as a *receive\_idle* event by an Application object. The behavior of an Application Object upon detection of the *receive\_idle* event is Application Object specific. A Bit–Strobe Command message that contains Application I/O Data is interpreted as a *run* event by an Application Object. The behavior of an Application Object upon detection of the *run* event is Application Object specific. See the Application Object descriptions in Volume II of the specification for more information concerning these events.

**Important:** Note that the CAN Data Field portion of the Bit–Strobe Command is either zero (0) or eight (8) bytes in length.

### 3-10.2 Bit–Strobe Response Message

The Bit–Strobe Response returns up to eight bytes of input data and/or status information to the Master from each Slave.

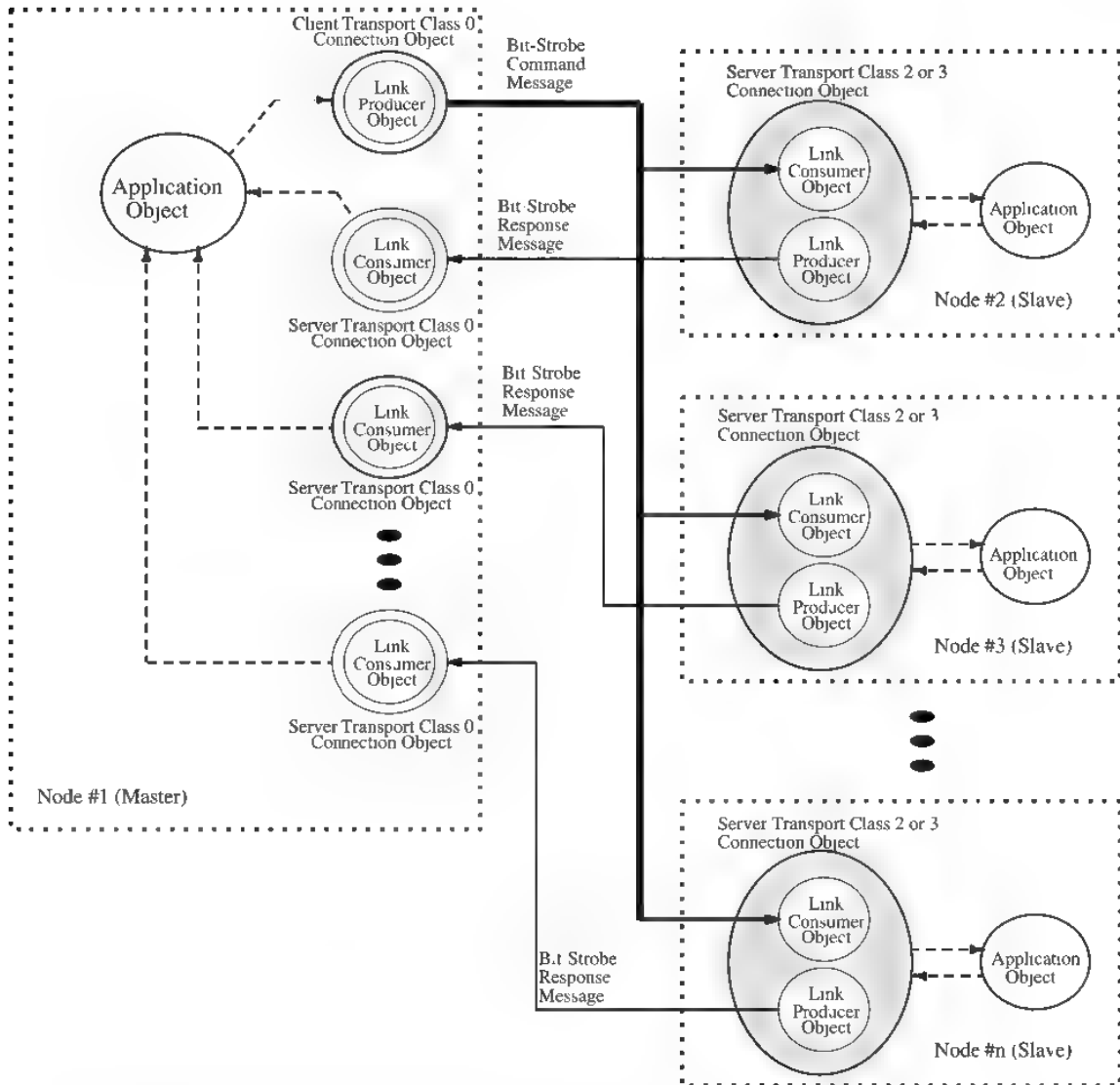
**Important:** The Bit–Strobe Connection was designed with the idea that it present a very efficient, fast mechanism for the exchange of a small amount of I/O between a Master and its Slaves. For this reason, the data field of a Bit–Strobe Response message may not exceed eight (8) bytes in length.

**Important:** A Bit–Strobe Response message that contains no Application I/O Data and was configured to contain data (indicated by the **produced\_connection\_size** attribute not equal to zero (0)) is defined to indicate a *no valid Bit–Strobe data for the master device* event. The master behavior upon detection of this event is vendor specific.

### 3-10.3 Bit–Strobe Message Characteristics

Figure 3-10.3 shows an overview of the Bit–Strobe relationship between a Master and its Slaves. The Master implements the communication resources associated with the Bit–Strobe Command as a Client Transport Class 0 Connection to transmit the command and a series of Server Transport Class 0 Connection Objects to receive the responses (1 Server Transport Class 0 Connection per Bit–Strobed Slave). The Slave implements a single Server Transport Class 2 or 3 Connection to receive the Bit–Strobe Command and send the associated response.

Figure 3-10.3 Bit–Strobe Connections



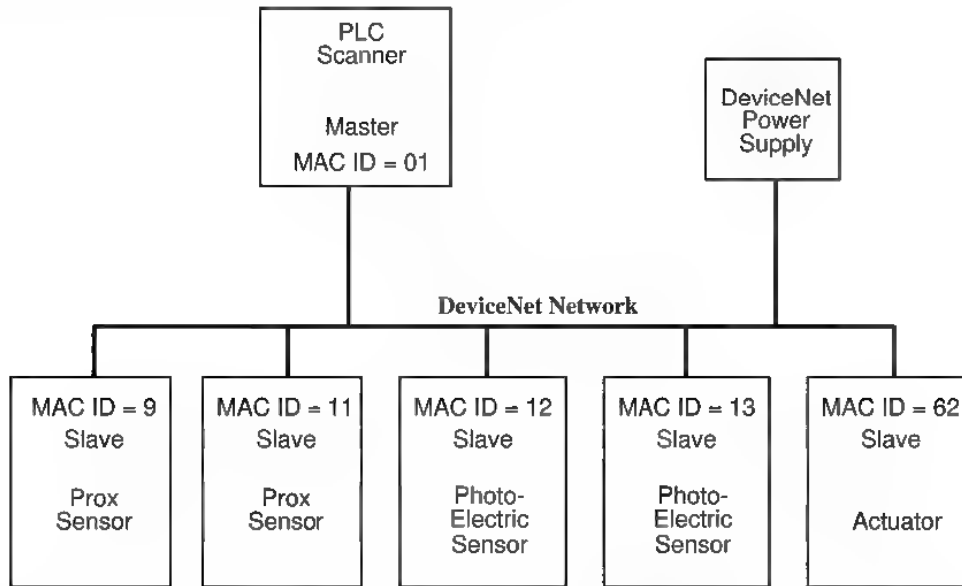
The process of allocating the Predefined Master/Slave Connection Set configures the Slave device to consume the multicast Bit–Strobe Command using its Bit–Strobe Connection, and informs the Slave of the Master's MAC ID. See section 0 for details of the Allocate\_Master/Slave\_Connection\_Set Service.

### 3-10.4 Bit–Strobe Example Application

In Figure 3-10.4 the Bit–Strobe application consists of one Master with five Slaves. The PLC Scanner (Master) wants to Bit–Strobe the sensors for input information and send output control data to an actuator. Figure 3-10.5 shows what I/O connections are needed to perform this application.



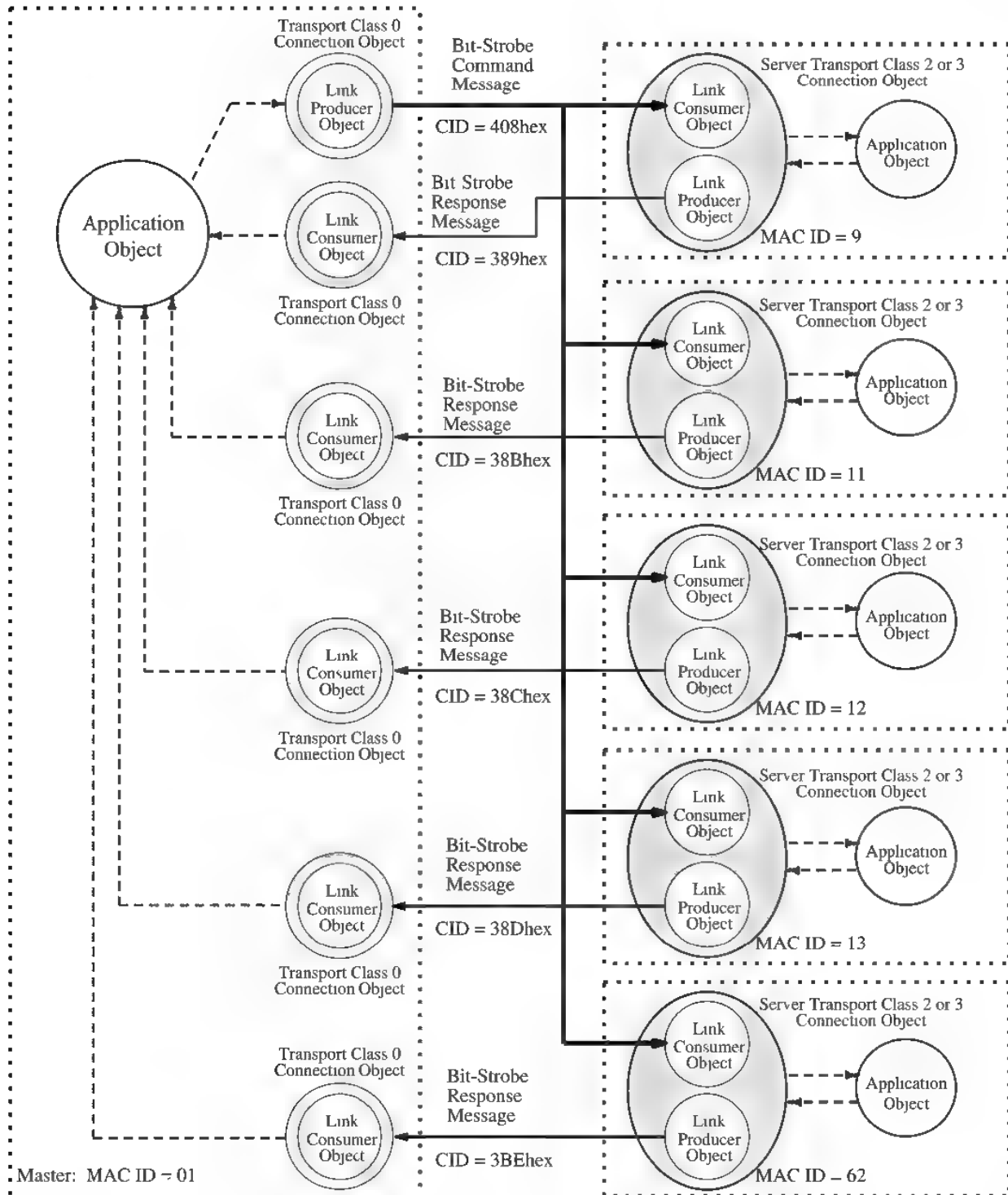
Figure 3-10.4 Bit-Strobe Example Application



All MAC IDs are shown as decimal numbers.

Figure 3-10.6, Table 3-10.1, and Table 3-10.2 show how the **produced\_connection\_id** and **consumed\_connection\_id** attributes (CIDs) for the Bit-Strobe Command and Response Connections between the Master and Slave with MAC ID 9 are determined.

Figure 3-10.5 Example of Bit-Strobe Connections



All MAC IDs are shown as decimal numbers. The CID is the value in the CAN Identifier Field and is shown in hex

The figure and tables below show how the **produced\_connection\_id** and **consumed\_connection\_id** attributes for the Bit-Strobe Connections within the Master whose MAC ID is 1 and the Slave whose MAC ID is 9 are determined. Included is a summary of bit values for the *Bit-Strobe Command message* (Table 3-10.1) and for one *Bit-Strobe Response message* (Table 3-10.2)

**Figure 3-10.6 Bit-Strobe Predefined Master/Slave Connection Identifier Fields**

IDENTIFIER BITS										IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1		
0	Group 1 Message ID				Source MAC ID					Group 1 Messages	000 – 3ff
0	1	1	1	0	Slave's MAC ID					Slave's I/O Bit Strobe Response Message	380 – 3bf
1	0	MAC ID				Group 2 Message ID				Group 2 Messages	400 – 5ff
1	0	Master's MAC ID					0	0	0	Master's I/O Bit-Strobe Command Message	400 – 5f8

**Table 3-10.1 CAN Identifier Bit Values for Bit-Strobe Command From MAC ID 1**

Because:	Bits	Hold the Value:
The Bit Strobe Command is a Group 2 message.	<b>10, 9</b>	10
The Master's MAC ID is 01.	<b>8 – 3</b>	000001
The Bit Strobe Command Message ID is 0.	<b>2 – 0</b>	000

A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Master's Client Bit-Strobe Connection in this example, the **produced\_connection\_id** would be set to 408<sub>hex</sub> which results in the bit pattern 100 0000 1000 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within each Slave's Bit-Strobe Connection is set accordingly. See Figure 3-10.5

**Table 3-10.2 CAN Identifier Bit Values for Bit-Strobe Response From MAC ID 9**

Because:	Bits	Hold the Value:
The Bit-Strobe Response is a Group 1 message.	<b>10</b>	0
The Bit Strobe Response Message ID is E.	<b>9 – 6</b>	1110
The Slave's MAC ID is 9.	<b>5 – 0</b>	001001

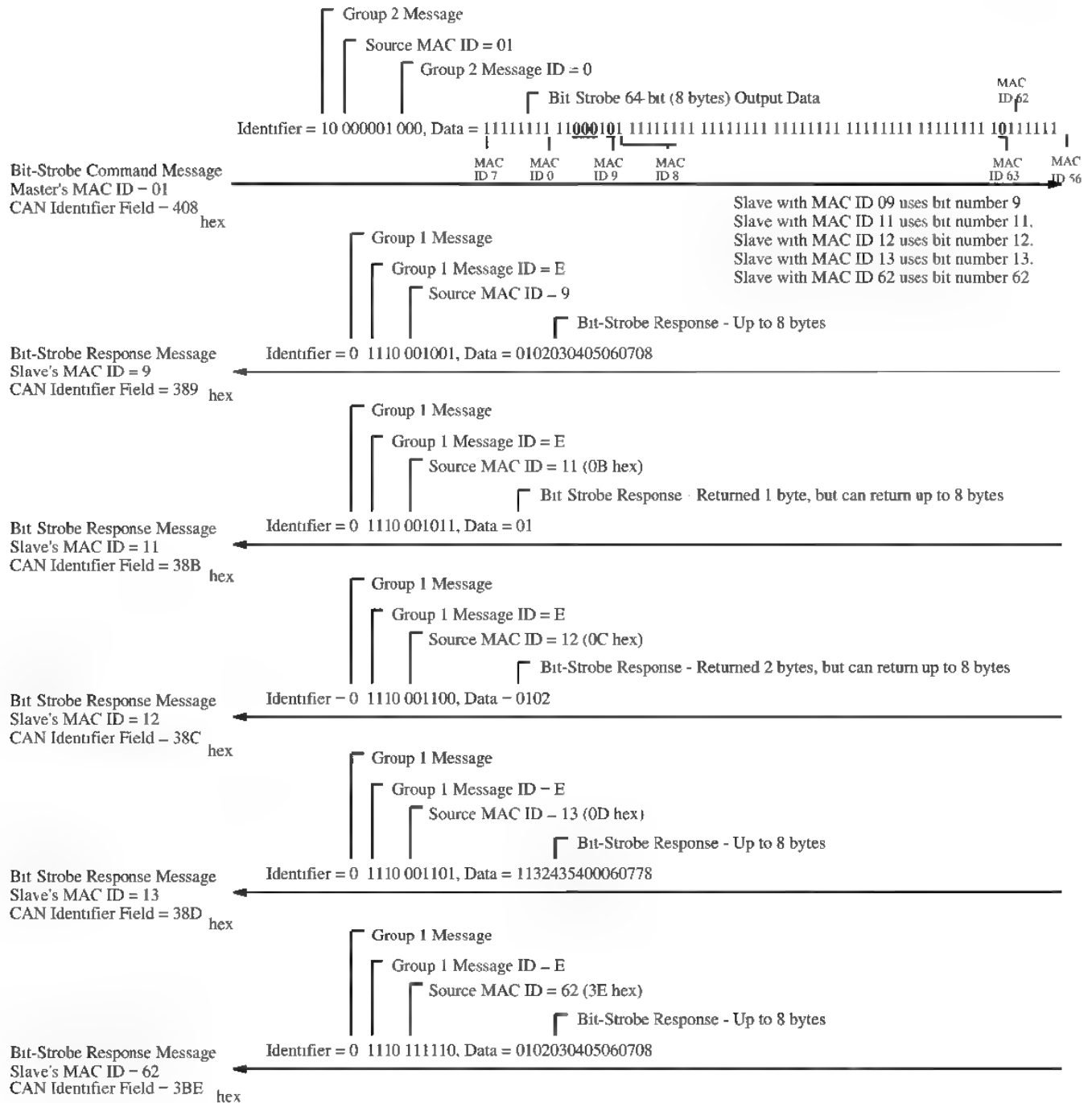
A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Slave's Bit-Strobe Connection in this example (MAC ID = 9), the **produced\_connection\_id** would be set to 389<sub>hex</sub> which results in the bit pattern 011 1000 1001 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within the Master's Server Connection Object that is responsible for receiving the Bit-Strobe response from MAC ID 9 is set accordingly. See Figure 3-10.5.

Figure 3-10.7 lists the Bit-Strobe Connection's **produced\_connection\_id** attribute for each Slave device. These are the values that are placed in the CAN Identifier Field when the Slave's Bit-Strobe Connection Object produces. Figure 3-10.8 shows example messages for this application.

**Figure 3-10.7 Slave Bit Strobe Connection produced\_connection\_id values for this example**

MAC ID (decimal)	MAC ID (hex)	MAC ID (binary)	produced_connection_id (binary)	produced_connection_id (hex)
9	09	00 1001	011 1000 1001	389
11	0B	00 1011	011 1000 1011	38B
12	0C	00 1100	011 1000 1100	38C
13	0D	00 1101	011 1000 1101	38D
62	3E	11 1110	011 1011 1110	3BE

Figure 3-10.8 Bit-Strobe Command and Response Example Messages



The slave's bits in the Strobe Command message are highlighted. The bits are shown as "0", but they can be "1" or "0"

### 3-11 Poll Command/Response Messages

Whereas the Bit-Strobe Command and Response messages move *small* amounts of I/O data between a Master and its Slaves, the Poll Command and Response messages move *any amount* of I/O data between a Master and its *Polled* Slaves.

#### 3-11.1 Poll Command Message

The **Poll Command** sends any amount of output data (non-fragmented or fragmented) to the destination Slave device.

A Slave device can be designed to do one or all of the following:

- Ignore the Poll Command (perhaps the Slave device is a Strobed device, or the Poll is not Allocated)
- Consume the Poll Command and its output data
- Consume the Poll Command as a trigger and ignore the output data

Slaves default to ignoring the Poll Command until the Poll Connection is allocated. See section 3-5 for information about allocation.

**Important:** *Not all DeviceNet devices are required to use poll output data.* For example, an input device can be engineered not to receive any Application I/O Data in the Poll Command, but use the Poll Command to trigger the transmission of return input data in a Poll Response message. In this case, the Master can transmit a zero byte Poll Command Message (no output data) to this slave.

**Important:** A Poll Command message transmitted with no Application I/O data in the CAN Data Field is interpreted as a *receive\_idle* event by an Application Object. The behavior of an Application Object upon detection of the *receive\_idle* event is Application Object specific. A Poll Command message that contains Application I/O Data is interpreted as a *run* event by an Application Object. The behavior of an Application Object upon detection of the *run* event is Application Object specific. Note that since the Poll Command can be fragmented, a "no data" Poll Command Message sent across a connection whose produced connection size is greater than 8 (fragmentation required) still contains the Fragmentation Protocol byte (first byte = 0x3f). See the Application Object descriptions in Volume II of the specification for more information concerning these events.

#### 3-11.2 Poll Response Message

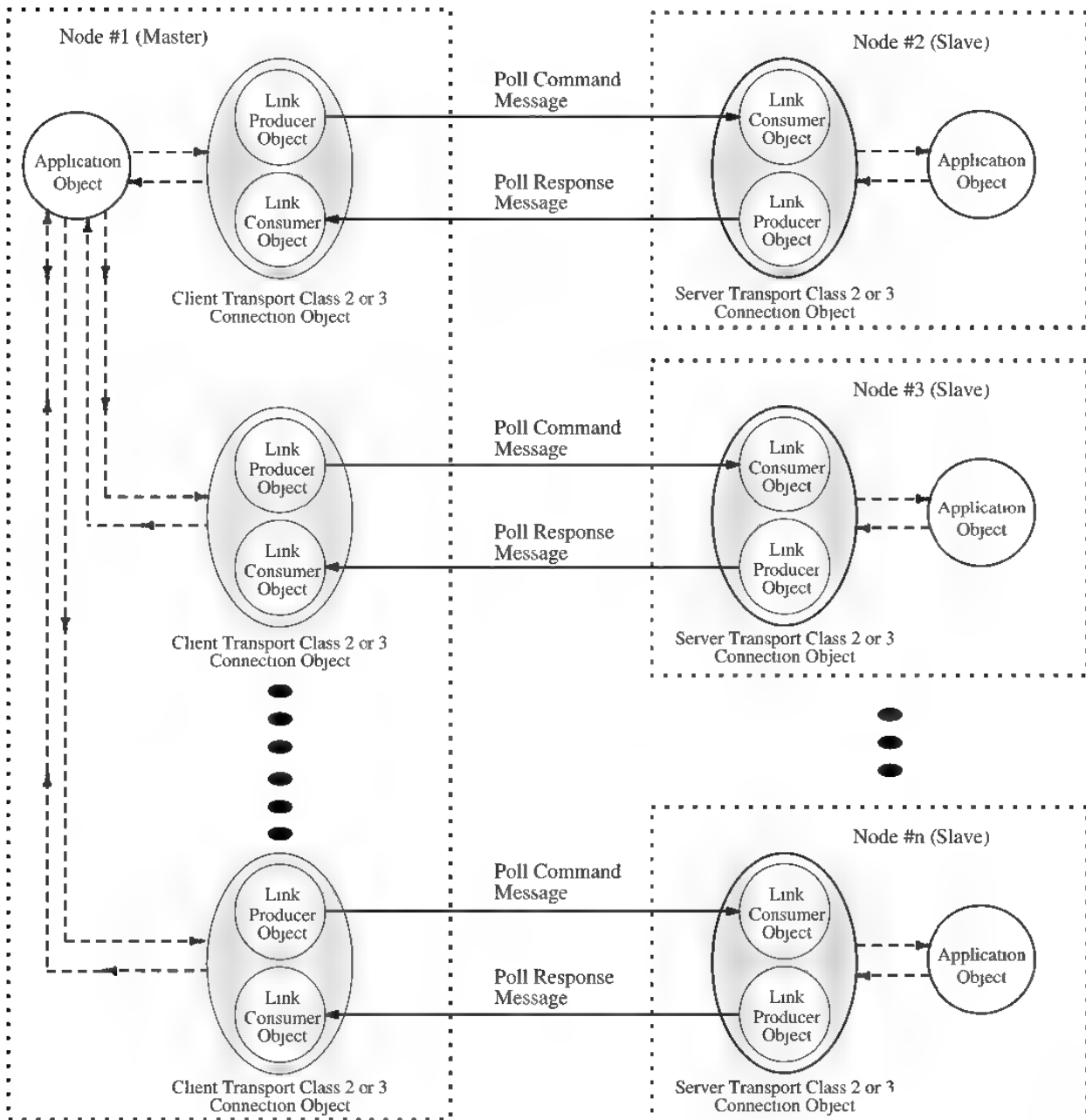
The **Poll Response** returns any amount (it can be non-fragmented or fragmented) of input data and/or status information to the Master from the Slave.

**Important:** A Poll Response message that contains no Application I/O Data and was configured to contain data (indicated by the **produced connection size** attribute not equal to zero (0)) is defined to indicate a *no valid Poll data for the master device* event. The master behavior upon detection of this event is vendor specific.

### 3-11.3 Poll Message Characteristics

Figure 3-11.1 shows an overview of the Poll relationship between a Master and its Slaves. The Master implements the communication resources associated with **each** Poll Command as a Client Transport Class 2 or 3 Connection Object. Each Slave implements a Server Transport Class 2 or 3 Connection Object to receive the Poll Command and send the associated response.

Figure 3-11.1 Poll Connections

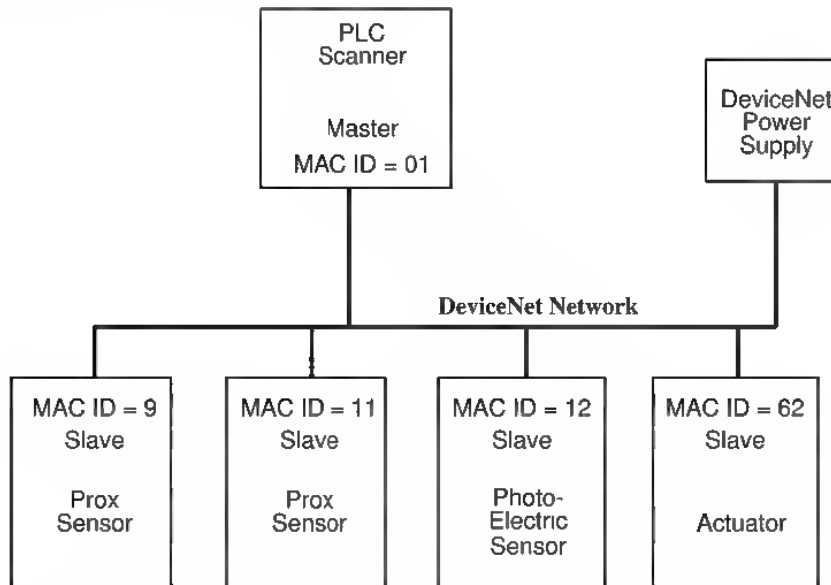


The process of allocating the `Predefined_Master/Slave_Connection_Set` configures the Slave device to consume the Poll Command and informs the Slave of the Master's MAC ID. See section 3-5 for details of the Allocate Service.

### 3-11.4 Poll Example Application

In Figure 3-11.2, the Poll application consists of one Master with four Slaves. The PLC Scanner (Master) wants to Poll the sensors for input information and send output control data to an actuator. Figure 3-11.3 shows what I/O connections are needed to perform this application.

Figure 3-11.2 Poll Example Application

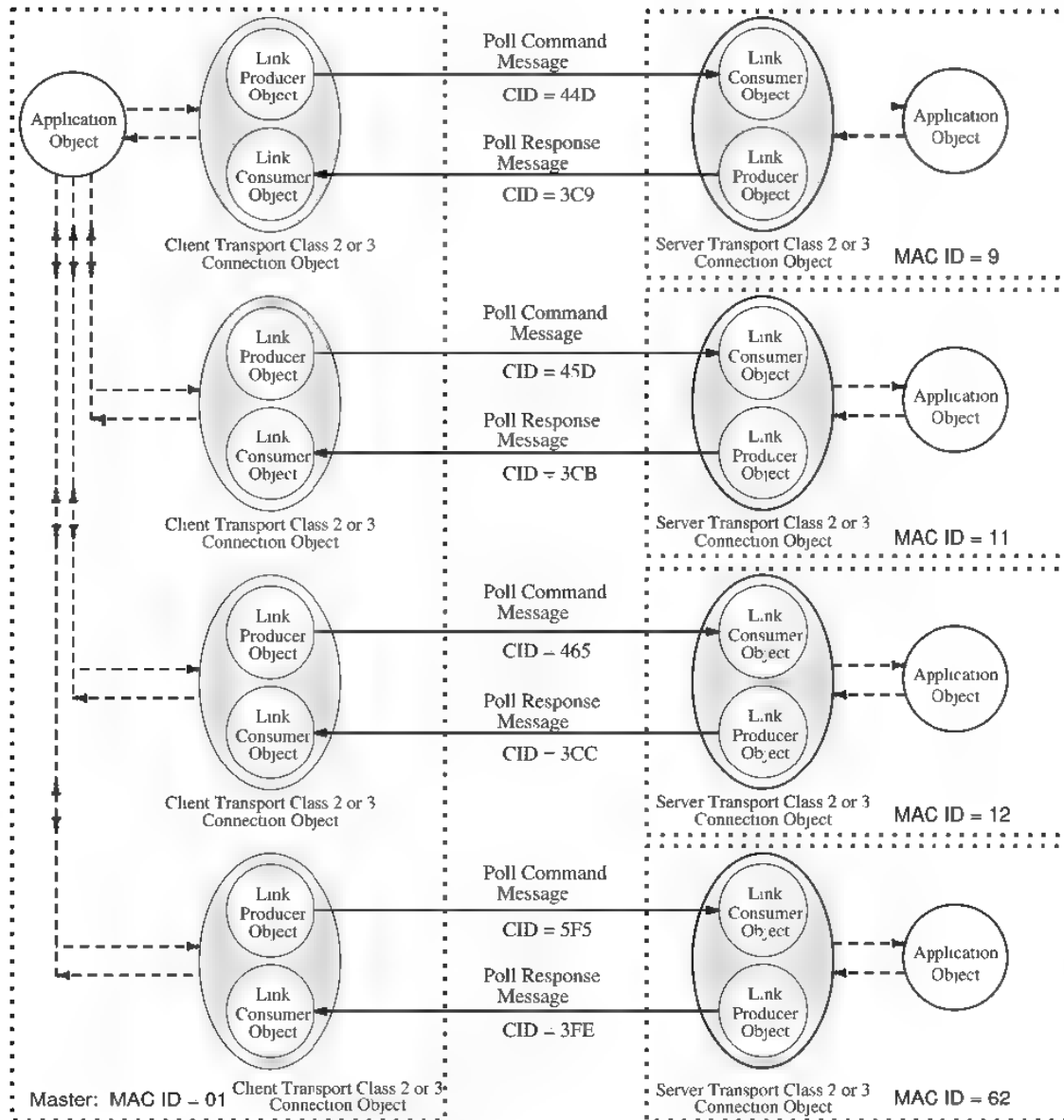


All MAC IDs are shown as decimal numbers.

Figure 3-11.4, Table 3-11.1 and Table 3-11.2 show how **produced\_connection\_id** and **consumed\_connection\_id** attributes (CIDs) for the Poll Command and Response Connections between the Master and Slave with MAC ID 9 are determined.



Figure 3-11.3 Example of Poll Connections



All MAC IDs are shown as decimal numbers  
The Connection ID (CID) is the value in the CAN Identifier Field and is shown in hex.

The figure and tables below show how the **produced\_connection\_id** and **consumed\_connection\_id** attributes for the Poll Connections within the Master and the Slave who's MAC ID is 9 are determined. Included is a summary of bit values for the *Poll Command message* (Table 3-11.1) and for the *Poll Response message* (Table 3-11.2) associated with it.

**Figure 3-11.4 Poll Predefined Master/Slave Identifier Fields**

IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MAC ID						Group 1 Messages	000 – 3ff
0	1	1	1	1	Slave's MAC ID						Slave's I/O Poll Response Message	3c0 – 3ff
1	0	MAC ID						Group 2 Message ID			Group 2 Messages	400 – 5ff
1	0	Slave's MAC ID						1	0	1	Master's I/O Poll Command Message	405 – 5fd

**Table 3-11.1 CAN Identifier Bit Values for Poll Command To MAC ID 9**

Because:	Bits	Hold the Value:
The Poll Command is a Group 2 message	<b>10, 9</b>	10
The Slave's MAC ID is 9.	<b>8 – 3</b>	001001
The Poll Command Message ID is 5.	<b>2 – 0</b>	101

A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Master's Poll Connection in this example, the **produced\_connection\_id** would be set to 44D<sub>hex</sub> which results in the bit pattern 100 0100 1101 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within each Slave's Poll Connection is set accordingly. See Figure 3-11.3.

**Table 3-11.2 CAN Identifier Bit Values for Poll Response From MAC ID 9**

Because:	Bits	Hold the Value:
The Poll Response is a Group 1 message.	<b>10</b>	0
The Poll Response Message ID is F.	<b>9 – 6</b>	1111
The Slave's MAC ID is 9.	<b>5 – 0</b>	001001

A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Slave's Poll Connection in this example, the **produced\_connection\_id** would be set to 3C9<sub>hex</sub> which results in the bit pattern 011 1100 1001 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within the Master's Poll Connection Object for this Slave is set accordingly. See Figure 3-11.3.

Table 3-11.3 lists the **produced\_connection\_id** attributes associated with both the Master's Poll Connections and the Slave's Poll Connections in this example. These are the values that are placed in the CAN Identifier Field when the Connection Objects produce data.

**Table 3-11.3 Master and Slave Poll Connection produced\_connection\_id attributes**

Slave MAC ID (decimal)	Slave MAC ID (hex)	Slave MAC ID (binary)	Master Poll Connection's produced_ connection_id attribute (binary)	Master Poll Connection's produced_ connection_id attribute (hex)	Slave Poll Connection's produced_ connection_id attribute (binary)	Slave Poll Connection's produced_ connection_id attribute (hex)
9	09	00 1001	100 0100 1101	44D	011 1100 1001	3C9
11	0B	00 1011	100 0101 1101	45D	011 1100 1011	3CB
12	0C	00 1100	100 0110 0101	465	011 1100 1100	3CC
62	3E	11 1110	101 1111 0101	5F5	011 1111 1110	3FE

Figure 3-11.5 Non-Fragmented Poll Command and Response Example Messages

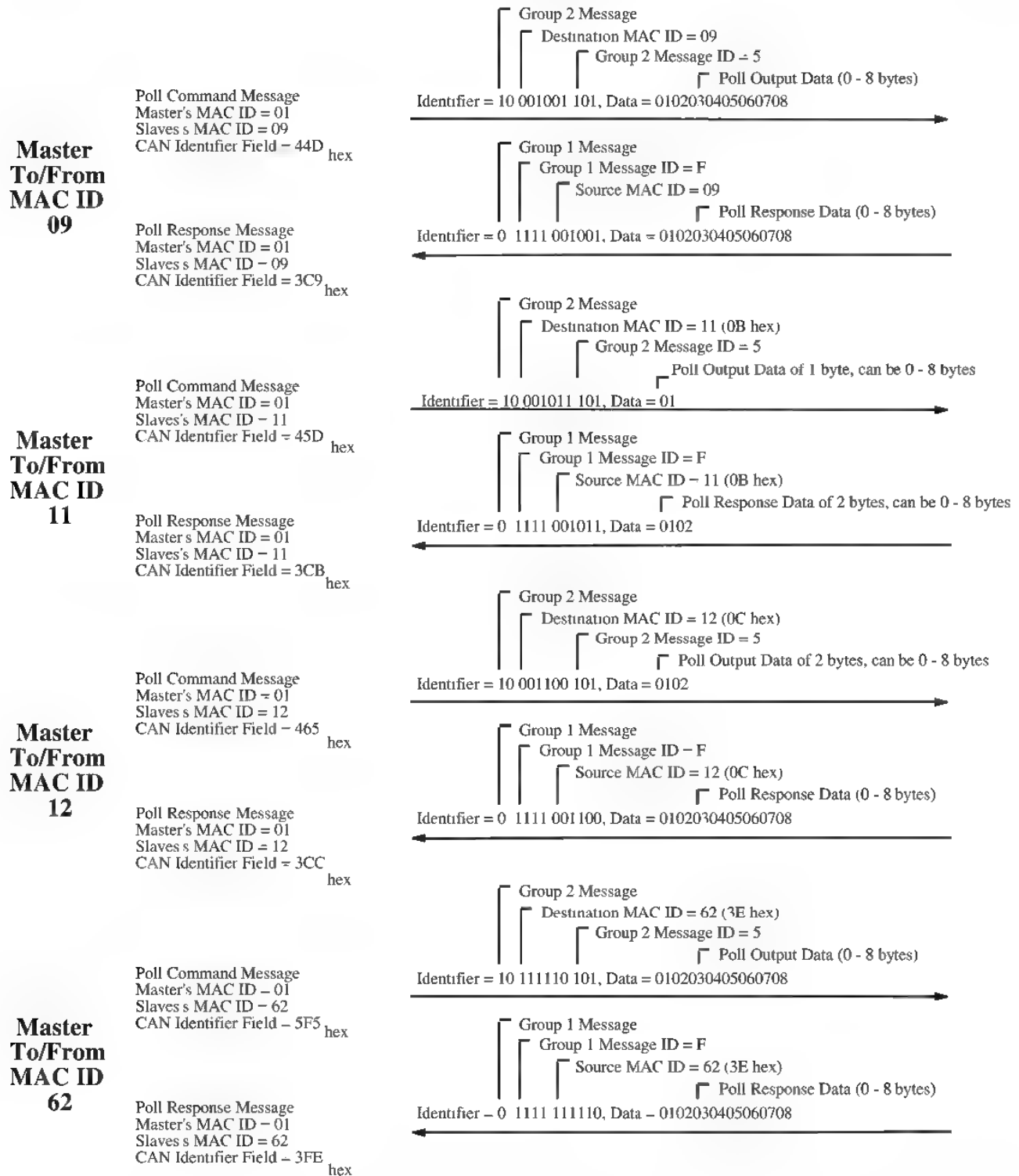
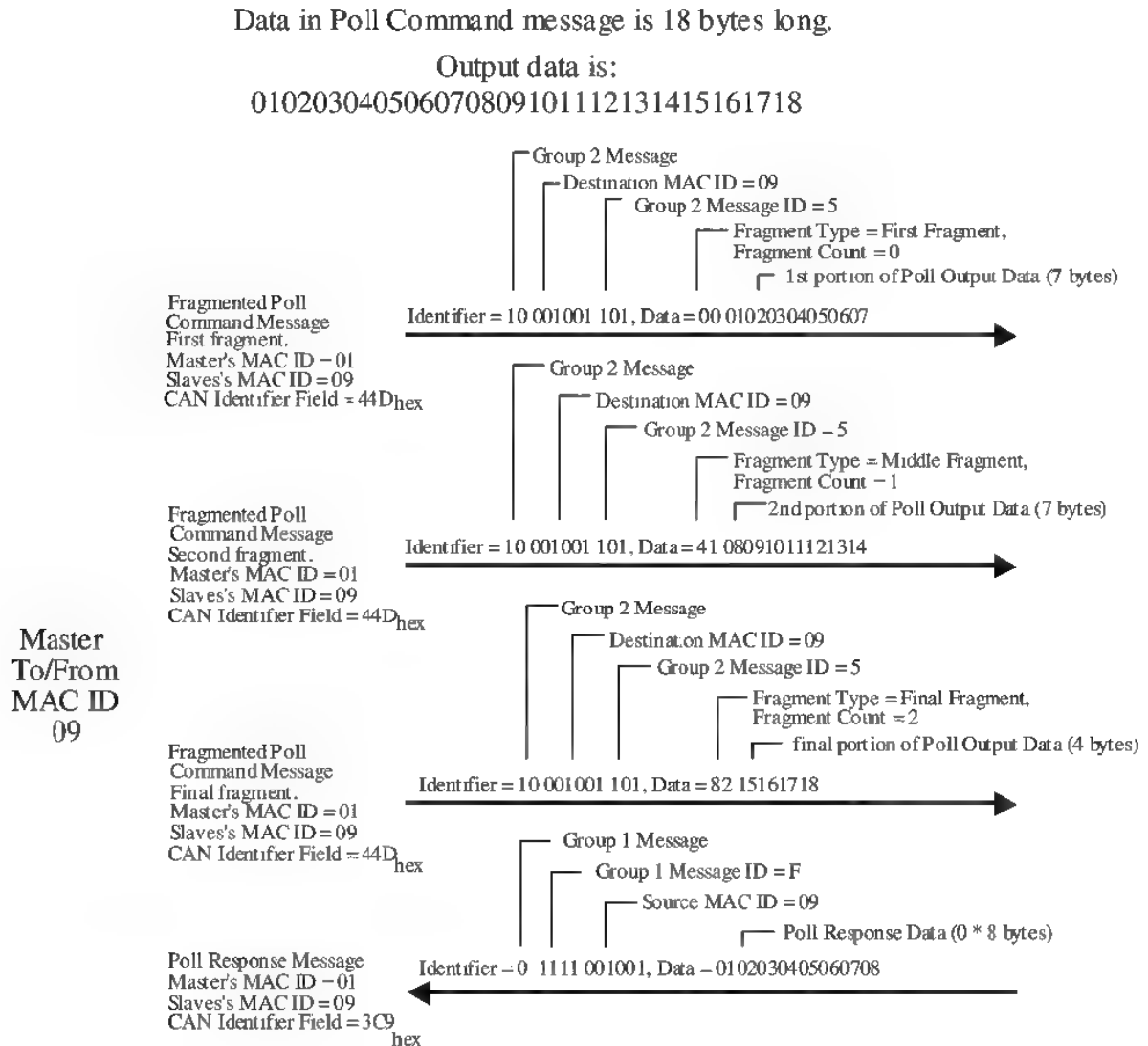


Figure 3-11.6 shows an example of a fragmented Poll Command message. In this example, the Master is sending 18 bytes of output data to slave with MAC ID 9. The Master has set the **connection\_size** attribute to 12<sub>hex</sub> (18 decimal). The Slave is responding with a non-fragmented Poll Response message.

**Figure 3-11.6 Fragmented Poll Command and Response Example Messages**



## 3-12 Multicast poll command/response messages

The Multicast Poll connection differs from the Poll connection in its ability to be multicast, rather than point-to-point. Any number of slaves may belong to a master's multicast group, and multiple multicast groups may exist for each master. This connection set allows for simultaneous sharing of data between the master and its multicast group.

### 3-12.1 Multicast Poll Command Message

The **Multicast Poll Command** *sends any amount of output data (non-fragmented or fragmented)* to the destination Slave device(s).

A Slave device can be designed to do one or all of the following:

- Ignore the Multicast Poll Command (perhaps the Slave device is a Polled device, or the Multicast Poll is not Allocated)
- Consume the Multicast Poll Command and its output data
- Consume the Multicast Poll Command as a trigger and ignore the output data

Slaves default to ignoring the Multicast Poll Command until the Multicast Poll Connection is allocated and established. See section 3-5 for information about allocation and connection establishment.

**Important:** *Not all DeviceNet devices are required to use Multicast Poll output data.*

For example, input devices can be engineered not to receive any Application I/O Data in the Multicast Poll Command, but use the Multicast Poll Command to trigger the transmission of input data in a Multicast Poll Response message. In this case, the Master can transmit a zero length Multicast Poll Command Message (no output data).

**Important:** A Multicast Poll Command message transmitted with no Application I/O data in the CAN Data Field is interpreted as a *receive\_idle* event by an Application Object. The behavior of an Application Object upon detection of the *receive\_idle* event is Application Object specific. A Multicast Poll Command message that contains Application I/O Data is interpreted as a *run* event by an Application Object. The behavior of an Application Object upon detection of the *run* event is Application Object specific.

The Multicast Poll Command message uses a Multicast MAC ID rather than a Destination MAC ID in the MAC ID field of the CAN identifier. The Multicast MAC ID shall be assigned by the master to a value corresponding to the MAC ID value of one of the slave members within the Multicast group or itself. When the multicast MAC ID is the master's MAC ID, it may only be used for one multicast group and precludes the master from being a slave in a multicast group to another master.

### 3-12.2 Multicast Poll Response Message

The **Multicast Poll Response** returns any amount (it can be non–fragmented or fragmented) of data to the Master from the Slave.

**Important:** A Multicast Poll Response message that contains no Application I/O Data and was configured to contain data (indicated by the **produced\_connection\_size** attribute not equal to zero (0)) is defined to indicate a *no data* event. The master behavior upon detection of this event is vendor specific.

### 3-12.3 Multicast Poll Message Characteristics

The default value for a slave's *consumed\_connection\_id* when allocated is 0xFFFF. The master shall perform a set to the *consumed\_connection\_id* attribute as defined in the following paragraphs before applying the connection attributes. A slave supporting this connection set shall allow a set to the *consumed\_connection\_id* attribute with a valid value. A slave shall not transition to established if a valid consumed connection identifier has not been set. A valid value for this attribute is Group 2, Message ID 1, any MAC ID. The response to the Multicast Poll message is made on Group 1 message identifier 12 and uses the slave's MAC ID as the source MAC ID.

The master assigns the multicast MAC ID after allocation by writing to the *consumed\_connection\_id* (Instance 5, Attribute 5 of the slave's Connection Object). Setting the *consumed\_connection\_id* may be done while the Multicast Poll slave's connection is in any state (except Non-Existent), and shall be set to a valid value before transitioning to established. If the application requires it, this multicast method can provide for a second standard poll I/O mechanism for any slave device that supports both Poll and Multicast Poll methods simultaneously.

After a Multicast poll message has been sent, all nodes allocated for that group respond on the group 1 "Slave's Multicast Response" identifier. Generation of the Multicast Poll request message to each group is application specific. A minimum of a zero length response (ack) is required from each device in order for the master to determine that the device (connection) is still present. The existing Poll, Strobe and COS/Cyclic connections can coexist with the Multicast Poll within the same device.

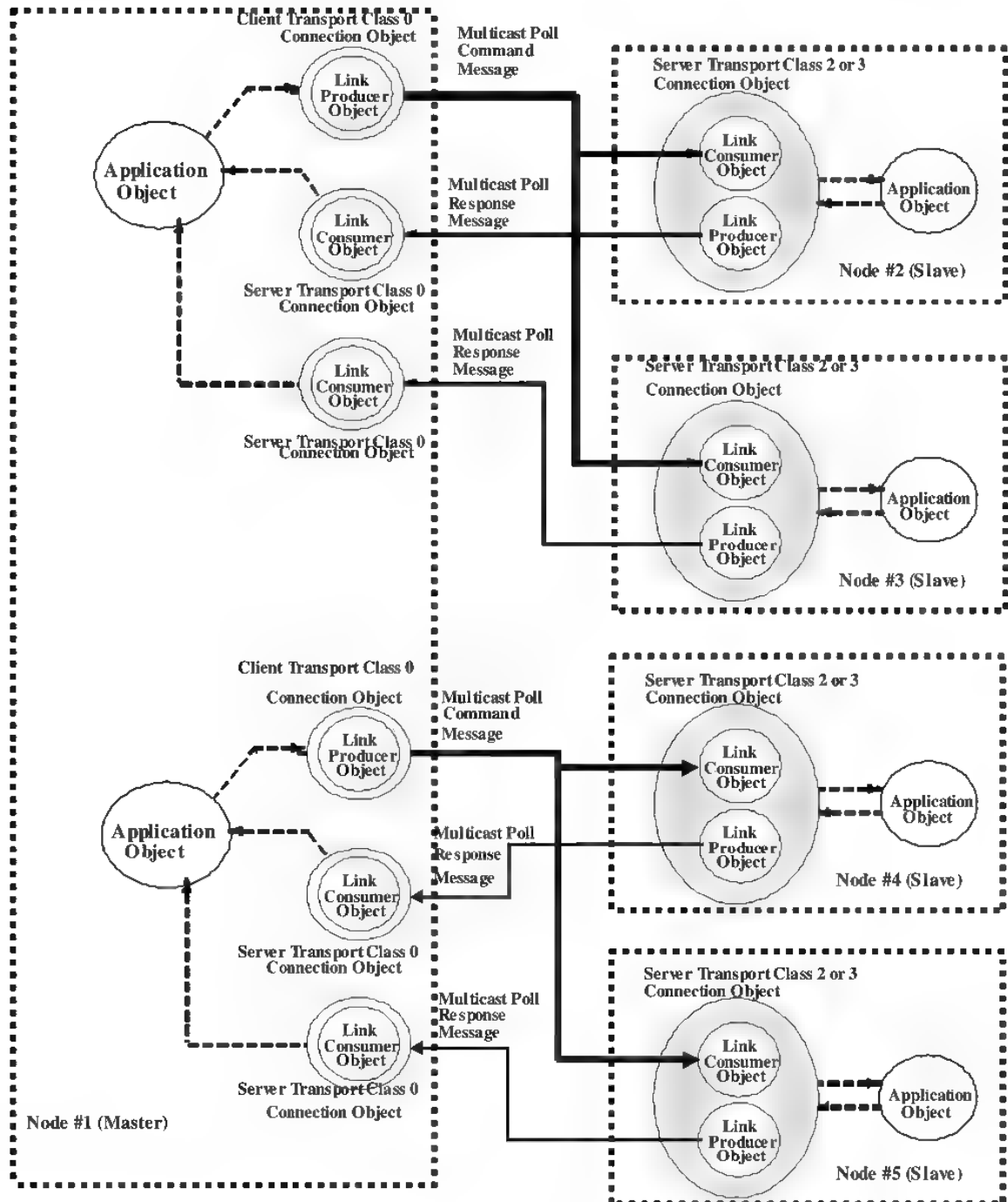
Release of Multicast Poll slaves operates like the release of the other predefined master/slave connection sets with the following exceptions. If a Master needs to release the slave whose MAC ID is being used for the Multicast MAC ID, the remaining Multicast slaves shall either have their *consumed\_connection\_id* attribute modified by the master or shall also be released. These actions shall occur prior to the release of the slave providing the Multicast MAC ID.

When the master detects the loss of a Multicast Poll connection to the slave device whose MAC ID is being used as the Multicast MAC ID the master shall:

- Transmit no more than one Multicast Poll Command message using the MAC ID of the lost slave.
- Choose a new MAC ID from a different slave in that multicast poll group to determine a new Connection Identifier.
- Send subsequent Multicast Poll Command messages with the new Connection Identifier.
- Modify the *consumed\_connection\_id* of the remaining slaves in that Multicast Poll group.

Figure 3-12.1 shows an overview of the Multicast Poll relationship between a Master and its Slaves. The Master implements the communication resources associated with each Multicast Poll Command as a Client Transport Class 0 Connection Object, and with each Multicast Poll Response as a Server Transport Class 0. Each Slave implements a Server Transport Class 2 or 3 Connection Object to receive the Multicast Poll Command and send the associated response.

Figure 3-12.1 Multicast Poll Connections

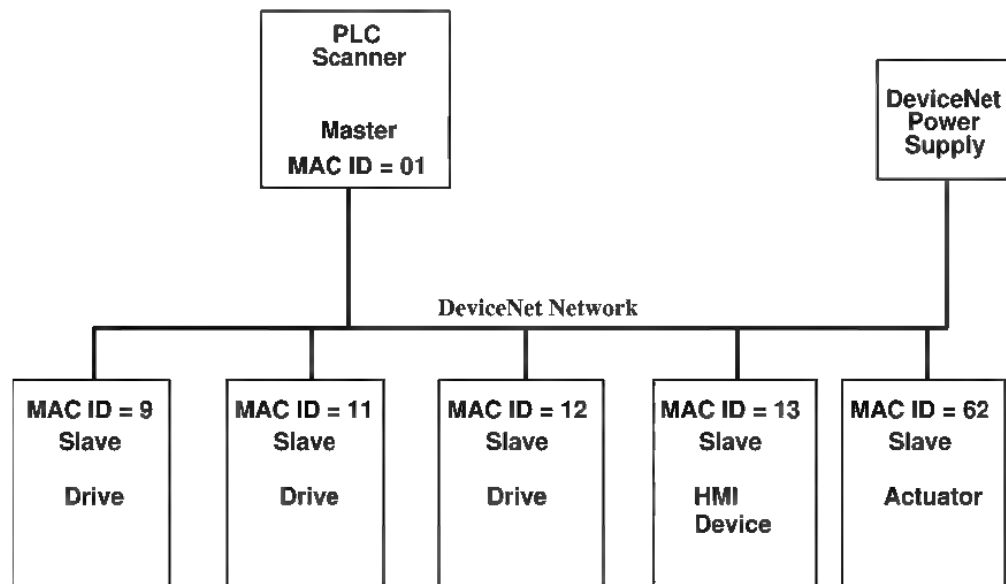




### 3-12.4 Multicast Poll Example Application

In Figure 3-12.2, the Multicast Poll application consists of one Master with five Slaves. The PLC Scanner (Master) sends Multicast Poll Command messages to the drives in one multicast group and to the HMI and Actuator devices in another multicast group. Figure 3-12.3 shows the I/O connections needed for this application.

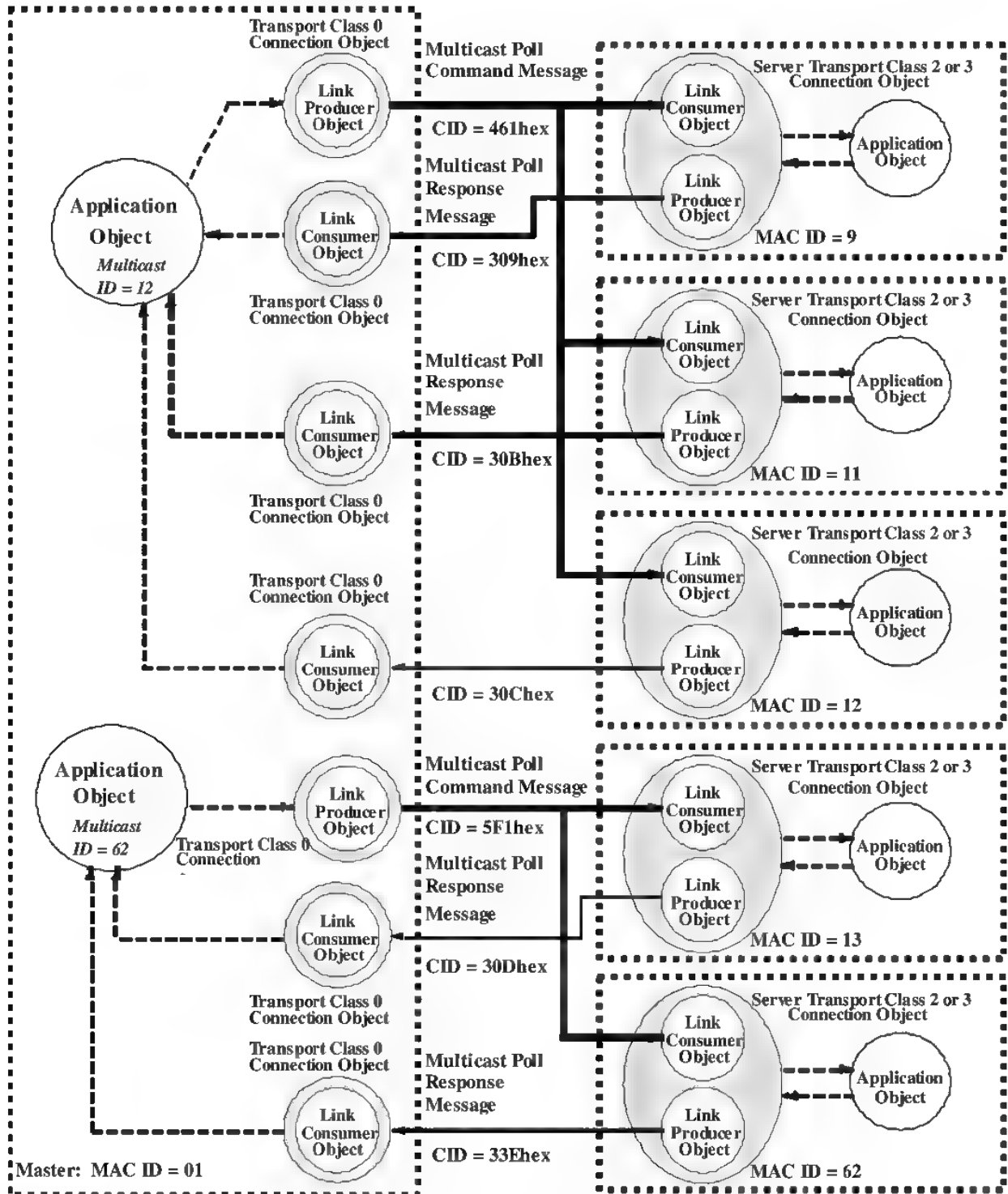
Figure 3-12.2 Multicast Poll Example Application



All MAC IDs are shown as decimal numbers.

Figure 3-12.4, Table 3-12.1 and Table 3-12.2 show how **produced\_connection\_id** and **consumed\_connection\_id** attributes (CIDs) for the Multicast Poll Command and Response Connections between the Master and Slave with MAC ID 9 are determined.

Figure 3-12.3 Example of Multicast Poll Connections



All MAC IDs are shown as decimal numbers. The CID is the value of the Connection Identifier.

The figure and tables below show how the **produced\_connection\_id** and **consumed\_connection\_id** attributes for the Multicast Poll Connections within the Master and the Slave whose MAC ID is 9 are determined. Included is a summary of bit values for the *Multicast Poll Command message* (Table 3-12.1) and for the *Multicast Poll Response message* (Table 3-12.2) associated with it.

**Figure 3-12.4 Multicast Poll Predefined Master/Slave Identifier Fields**

IDENTIFIER BITS										IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1		
0	Group 1 Message ID				Source MAC ID					Group 1 Messages	000 – 3ff
0	1	1	0	0	Slave's MAC ID					Slave's I/O Multicast Poll Response Message	300 – 33f
1	0	MAC ID				Group 2 Message ID				Group 2 Messages	400 – 5ff
1	0	Master's MAC ID					0	0	1	Master's I/O Multicast Poll Command Message	401 – 5f9

**Table 3-12.1 CAN Identifier Bit Values for Multicast Poll Command To MAC ID 9**

Because:	Bits	Hold the Value:
The Multicast Poll Command is a Group 2 message.	<b>10, 9</b>	10
The Multicast MAC ID is 12.	<b>8 – 3</b>	001100
The Multicast Poll Command Message ID is 1.	<b>2 – 0</b>	001

A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Master's Multicast Poll Connection in this example, the **produced\_connection\_id** would be set to 461<sub>hex</sub> which results in the bit pattern 100 0110 0001 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within each Slave's Multicast Poll Connection is set accordingly. See Figure 3-12.3.

**Table 3-12.2 CAN Identifier Bit Values for Multicast Poll Response From MAC ID**

Because:	Bits	Hold the Value:
The Multicast Poll Response is a Group 1 message	<b>10</b>	0
The Multicast Poll Response Message ID is 12.	<b>9 – 6</b>	1100
The Slave's MAC ID is 9.	<b>5 – 0</b>	001001

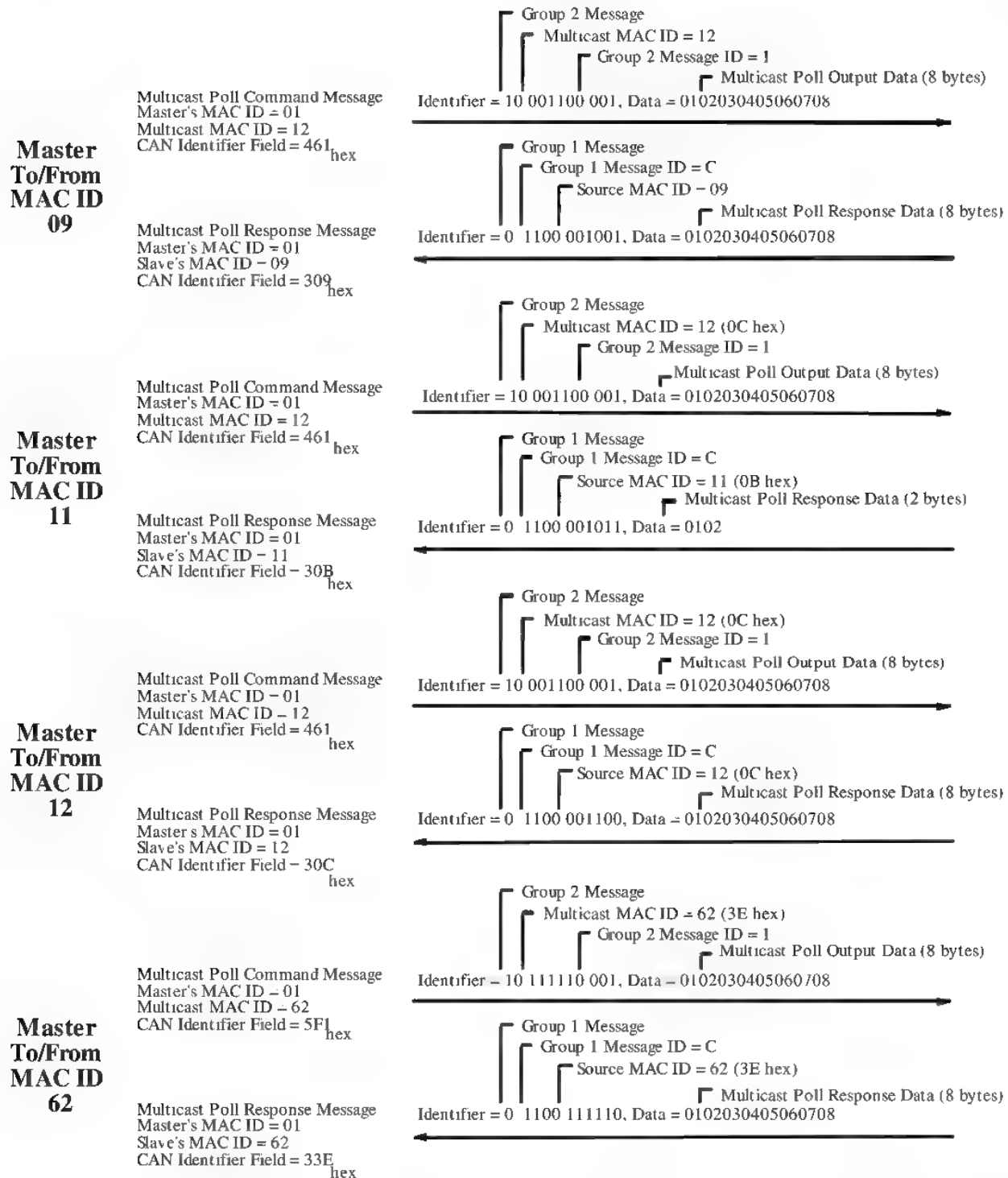
A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Slave's Multicast Poll Connection in this example, the **produced\_connection\_id** would be set to 309<sub>hex</sub> which results in the bit pattern 011 0000 1001 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within the Master's Multicast Poll Connection Object for this Slave is set accordingly. See Figure 3-12.3.

Table 3-12.3 lists the **produced\_connection\_id** attributes associated with both the Master's Multicast Poll Connections and the Slaves Multicast Poll Connections in this example. These are the values that are placed in the CAN Identifier Field when the Connection Objects produce data. Figure 3-12.5 shows non-fragmented example messages for this application.

**Table 3-12.3 Master and Slave Multicast Poll Connection produced\_connection\_id attributes**

<b>Multicast MAC ID (decimal)</b>	<b>Multicast MAC ID (hex)</b>	<b>Multicast MAC ID (binary)</b>	<b>Slave MAC ID (decimal)</b>	<b>Slave MAC ID (hex)</b>	<b>Slave MAC ID (binary)</b>	<b>Master Multicast Poll Connection's produced connection_id attribute (binary)</b>	<b>Master Multicast Poll Connection's produced connection_id attribute (hex)</b>	<b>Slave Multicast Poll Connection's produced connection_id attribute (binary)</b>	<b>Slave Multicast Poll Connection's produced connection_id attribute (hex)</b>
12	0C	00 1100	9	09	00 1001	100 0110 0001	461	011 0000 1001	309
12	0C	00 1100	11	0B	00 1011	100 0110 0001	461	011 0000 1011	30B
12	0C	00 1100	12	0C	00 1100	100 0110 0001	461	011 0000 1100	30C
62	3E	11 1110	13	0D	00 1101	101 1110 0001	5F1	011 0000 1101	30D
62	3E	11 1110	62	3E	11 1110	101 1110 0001	5F1	011 0011 1110	33E

Figure 3-12.5 Non-Fragmented Multicast Poll Command and Response Example Messages

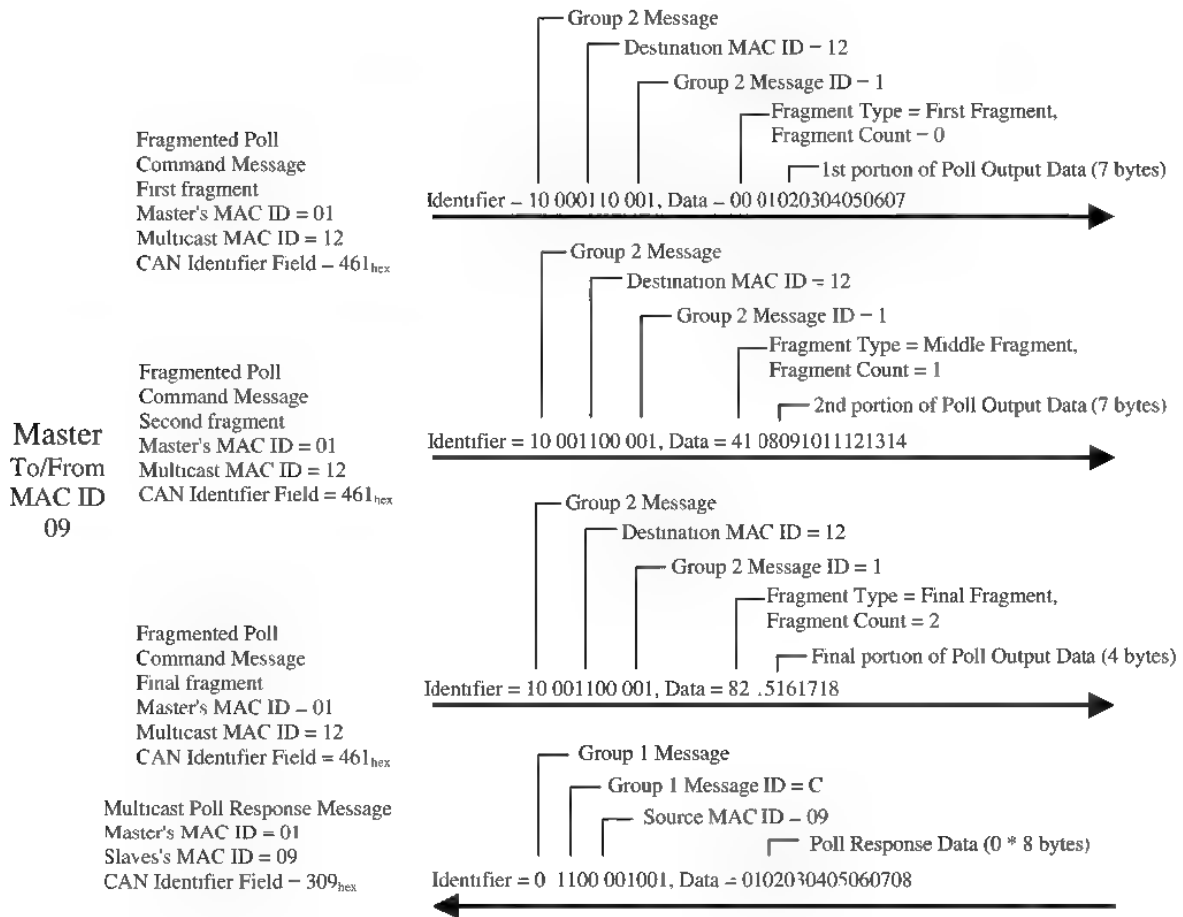


Note: Node 13 not shown

Figure 3-12.6 shows an example of a fragmented Multicast Poll Command message. In this example the Master is sending 18 bytes of output data to the slaves at MAC IDs 9, 11, and 12. The Master has set the **connection size** attribute to 12<sub>hex</sub> (18 decimal). The Slave is responding with a non-fragmented Multicast Poll Response message. Only the response from the node at MAC ID 9 is shown.

**Figure 3-12.6 Fragmented Multicast Poll Command and Response Example Messages**

Data in Multicast Poll Command message is 18 bytes long.  
Output data is: 010203040506070809101112131415161718



### **3-13 Change of State/Cyclic Messages**

The Predefined Master/Slave Connection Set will now support Change of State or Cyclic data production between the master and the slave. This data production can be either acknowledged or unacknowledged.

The Change of State/Cyclic connection sets use connection instance 2 (the polled connection instance) for master to slave data production and slave to master acknowledgment. Connection instance 4 (previously reserved) is used for slave to master data production and master to slave acknowledgment. If a device does not support the poll, and has no support for output data, connection instance 2 does not need to be instantiated.

To keep the system behavior consistent, the slave's Change of State/Cyclic connection (instance 4) must have the consumed path set the Acknowledge Handler Object. Furthermore, the instance of the Acknowledge Handler Object must be one.

Because the Polled and Change of State/Cyclic connection sets share connection instance 2, the slave must follow certain procedures based on the allocation request to insure the correct behavior. As described above, when only the Change of State/Cyclic allocation bit is on connection instances 2 and 4 are instantiated. Connection instance 4 produces from the default input path and consume acknowledgments to instance 1 of the Acknowledge Handler Object. Connection instance 2 consumes to the default output path and produces a zero length acknowledgment. When the allocation request contains only the Polled allocation bit, connection instance 2 consumes to the default output path and produces from the default input path.

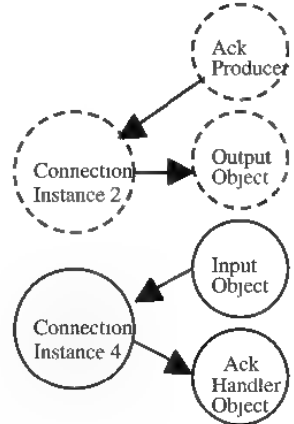
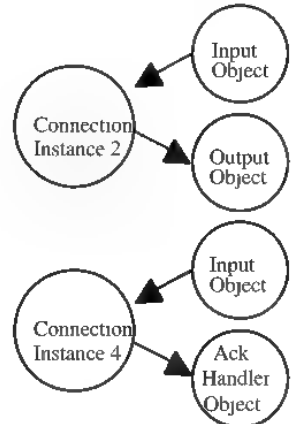
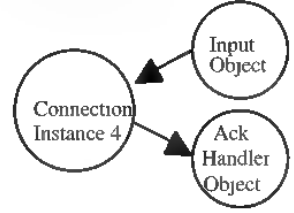
If both the Change of State/Cyclic *and* the Polled allocation bits are on, connection instance 2 will behave as though only the polled allocation bit was on, and connection instance 4 will continue to behave as described above. This combination is useful when a master wants to poll the slave for some inputs, and receive different inputs change of state or cyclically.

This would require the two default input paths to have been previously configured at the slave, or at the master if it will set these paths before activating the connections. This is also the behavior if the master allocates the Poll connection set in one message, and then the Change of State/Cyclic connection set in a following message.

To achieve unacknowledged data production, the Acknowledge Suppression bit is set with either the Change of State or Cyclic bit in the Allocation Choice byte. Connection Instance 2 is configured as a class 0 connection for master to slave data production. Connection 4 is configured as a class 0 connection for slave to master data production. The exception to this is when the Poll bit is set in the same allocation choice, which causes Connection Instance 2 to be configured as the Poll, while Connection Instance 4 continues to be configured as an Unacknowledged Change of State or Cyclic connection.

The following scenarios are possible for both acknowledged and unacknowledged connections (ignoring the strobe and explicit messaging connection sets which are unaffected by these additions).

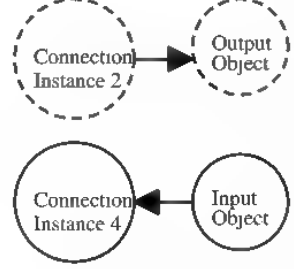
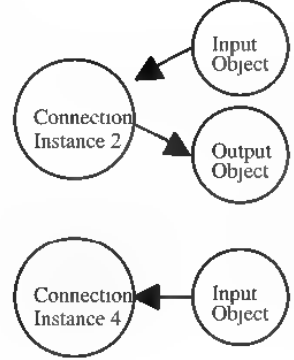
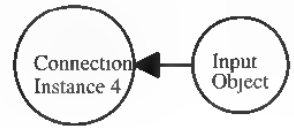
Figure 3-13.1 Acknowledged Change of State

Allocation Choice	Slave Behavior	Acknowledged Change of State/Cyclic
COS	Slave instantiates connection instances 2 and 4. Instance 2 is configured as a Class 2/3 Server. The produced connection size is set to zero and the produced connection path is NULL or an Acknowledge Producing application object. The consumed connection path is set to the default output path. Instance 4 is configured as a Class 2/3 Client, Change of State triggered. The produced connection path is set to the default input path. The consumed path is set to the Acknowledge Handler Object. If a device has no outputs and does not support the poll connection, then instance 2 does not need to be instantiated.	
COS + Poll	Slave instantiates connection instances 2 and 4. Instance 2 is configured as a Class 2/3 Server. The produced connection size and produced connection path are left at the default input path (which may have been previously configured). The consumed connection path is set to the default output path. Instance 4 is configured as a Class 2/3 Client, Change of State triggered. The produced connection path is set to the default input path. The consumed path is set to the Acknowledge Handler Object.	
Poll, then COS on separate allocation message.	Slave instantiates connection instance 2 and is configured as a Class 2/3 Server. The produced connection size and produced connection path are set to the default input path (which may have been previously configured). The consumed connection path is set to the default output path. On the following allocation message, Instance 4 is instantiated and configured as a Class 2/3 Client, Change of State triggered. The produced connection path is set to the default input path. The consumed path is set to the Acknowledge Handler Object.	
COS, then Poll on separate allocation message	The slave will behave as described above for the COS allocation. The Poll allocation message which follows will be rejected. An error will indicate Resource Unavailable (02h) with an additional error code of 04h.	

**Note:** The Cyclic connection set has the same allocation behavior as the Change of State Connection Set illustrated above. Since these two connection sets are mutually exclusive, the word ‘Cyclic’ can be substituted where ‘COS’ and ‘Change of State’ appear.



Figure 3-13.2 Unacknowledged Change of State

Allocation Choice	Slave Behavior	Unacknowledged Change of State/Cyclic
COS + Acknowledge Suppression	Slave instantiates connection instances 2 and 4. Instance 2 is configured as a Class 0 Server. The produced connection size and the produced connection path are not configured. The consumed connection path is set to the default output path. Instance 4 is configured as a Class 0 Client, Change of State triggered. The produced connection path is set to the default input path. The consumed path is not configured. If a device has no outputs and does not support the poll connection, then instance 2 does not need to be instantiated.	
COS + Poll + Acknowledge Suppression	Slave instantiates connection instances 2 and 4. Instance 2 is configured as a Class 2/3 Server. The produced connection size and produced connection path are left at the default input path (which may have been previously configured). The consumed connection path is set to the default output path. Instance 4 is configured as a Class 0 Client, Change of State triggered. The produced connection path is set to the default input path. The consumed path is not configured.	
Poll, then COS + Acknowledge Suppression on separate allocation message.	Slave instantiates connection instance 2 and is configured as a Class 2/3 Server. The produced connection size and produced connection path are set to the default input path (which may have been previously configured). The consumed connection path is set to the default output path. On the following allocation message, Instance 4 is instantiated and configured as a Class 0 Client, Change of State triggered. The produced connection path is set to the default input path. The consumed path is not configured.	
COS + Acknowledge Suppression, then Poll on separate al location message	The slave will behave as described above for the COS allocation. The Poll allocation message which follows will be rejected. An error will indicate Resource Unavailable (02h) with an additional error code of 04h.	

**Note:** The Cyclic connection set has the same allocation behavior as the Change of State Connection Set as illustrated above. Since these two connection sets are mutually exclusive, the word 'Cyclic' can be substituted where 'COS' and 'Change of State' appear.

### 3-13.1 Change of State/Cyclic Messages

The Change of State/Cyclic messages move *any amount* of I/O data between a Master and Slave using Change of State or Cyclic production triggers. Data production can be either acknowledged or unacknowledged.

This section contains the following information pertaining to Change of State/Cyclic messages:

- Change of State/Cyclic Message (Acknowledged)
- Change of State/Cyclic Message (Unacknowledged)
- Change of State/Cyclic Message Characteristics
- Change of State/Cyclic Example Application

### 3-13.2 Master's Change of State/Cyclic Message

The *Master's Change of State/Cyclic Message* sends any amount of output data (non-fragmented or fragmented) to the destination Slave device. Data production is triggered by either a Change of State or transmission trigger expiration.

**Important:** A Change of State/Cyclic message transmitted with no Application I/O data in the CAN Data Field is interpreted as a *receive\_idle* event by an Application Object. The behavior of an Application Object upon detection of the *receive\_idle* event is Application Object specific. A Change of State/Cyclic message that contains Application I/O Data is interpreted as a *run* event by an Application Object. The behavior of an Application Object upon detection of the *run* event is Application Object specific. Note that since the Change of State/Cyclic can be fragmented, a "*no data*" Change of State/Cyclic Message sent across a connection whose produced connection size is greater than 8 (fragmentation required) still contains the Fragmentation Protocol byte (first byte = 0x0f). See the Application Object descriptions in Volume 1, Chapter 5 for more information concerning these events.

### 3-13.3 Slave's Change of State/Cyclic Acknowledge Message

The *Slave's Change of State/Cyclic Acknowledge Message* returns any amount (it can be non-fragmented or fragmented) of input data and/or status information to the Master from the Slave. By default, the acknowledge message is zero length.

### 3-13.4 Slave's Change of State/Cyclic Message

The *Slave's Change of State/Cyclic Message* sends any amount of output data (non-fragmented or fragmented) to the Master from the Slave. Data production is triggered by either Change of State or transmitted trigger expiration.

**Important:** A Change of State/Cyclic message that contains no Application I/O Data and was configured to contain data (indicated by the **produced\_connection\_size** attribute not equal to zero (0)) is defined to indicate a *no valid Change of State/Cyclic data for the master device* event. The master behavior upon detection of this event is vendor specific.

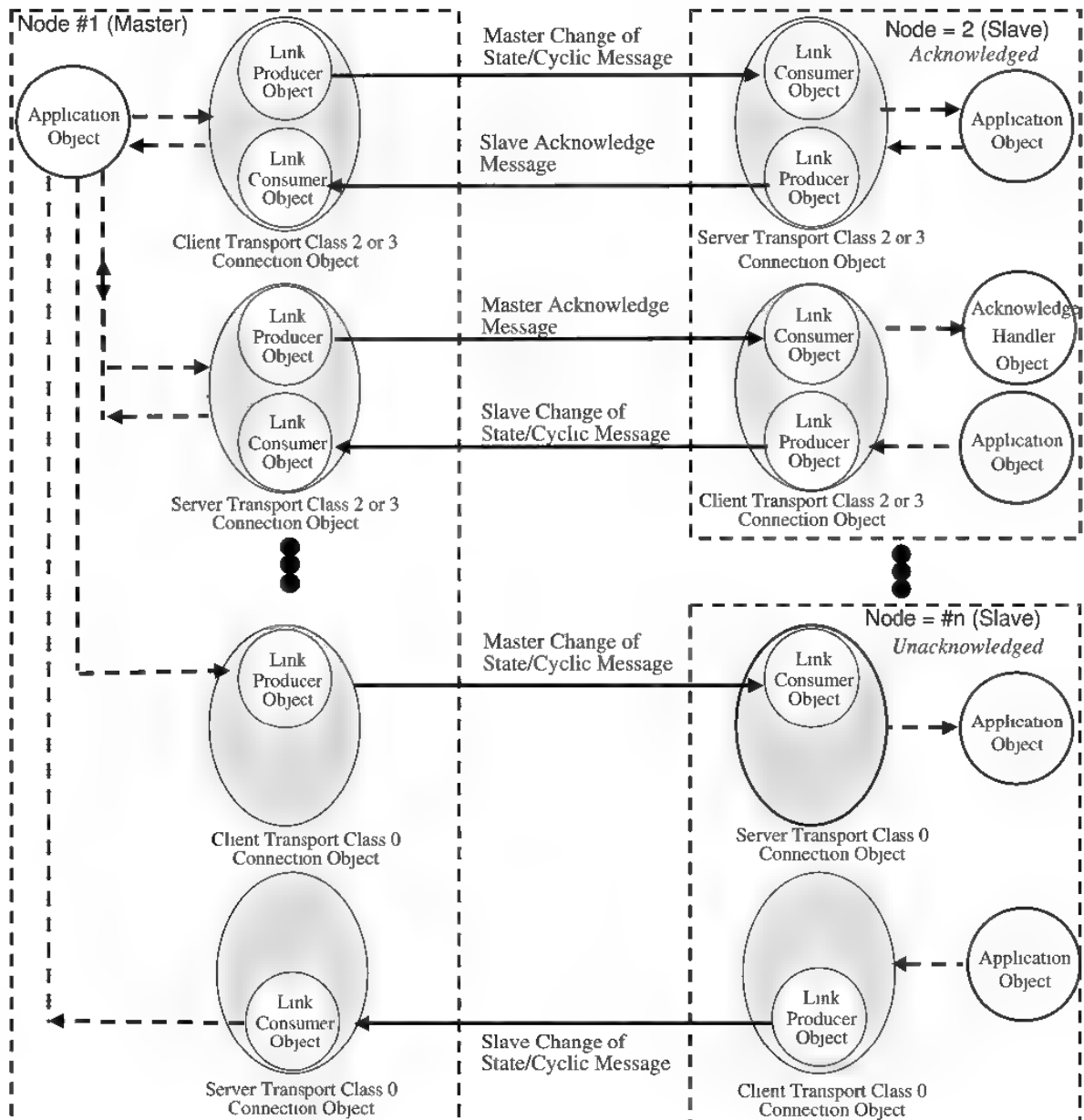
### 3-13.5 Master's Change of State/Cyclic Acknowledge Message

The *Slave's Change of State/Cyclic Acknowledge Message* returns any amount (it can be non-fragmented or fragmented) of input data and/or status information to the Slave from the Master. By default, the acknowledge message is zero length.

### 3-13.6 Change of State/Cyclic Message Characteristics

Figure 3-13.3 shows an overview of the Change of State/Cyclic message connections. All of the connections are point-to-point.

Figure 3-13.3 Change of State/Cyclic Connections

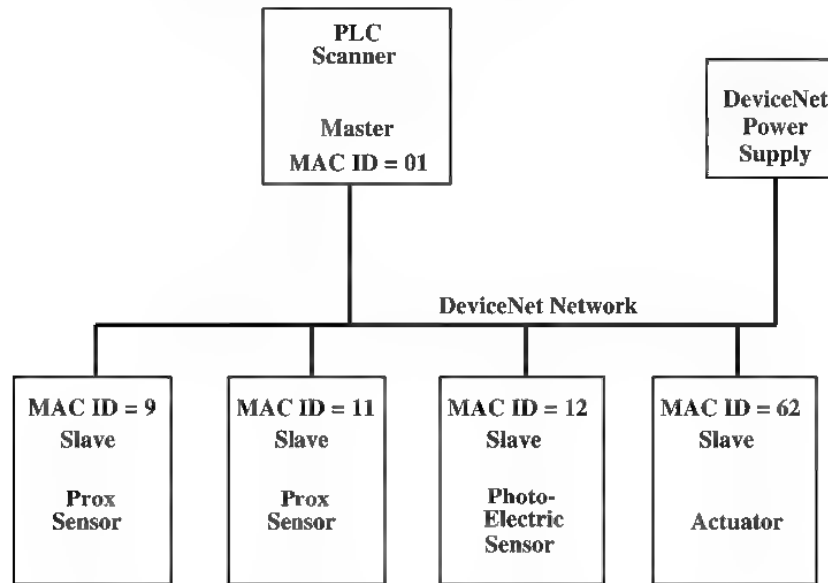


The process of allocating the `Predefined_Master/Slave_Connection_Set` configures the Slave device to “hear” the Change of State/Cyclic messages and informs the Slave of the Master’s MAC ID. See section 0 for details of the Allocate Service.

### 3-13.7 Change of State/Cyclic Example Application

In Figure 3-13.4, the Change of State/Cyclic application consists of one Master with four Slaves. Figure 3-13.5 shows what I/O connections are needed to perform this application.

Figure 3-13.4 Change of State/Cyclic Example Application

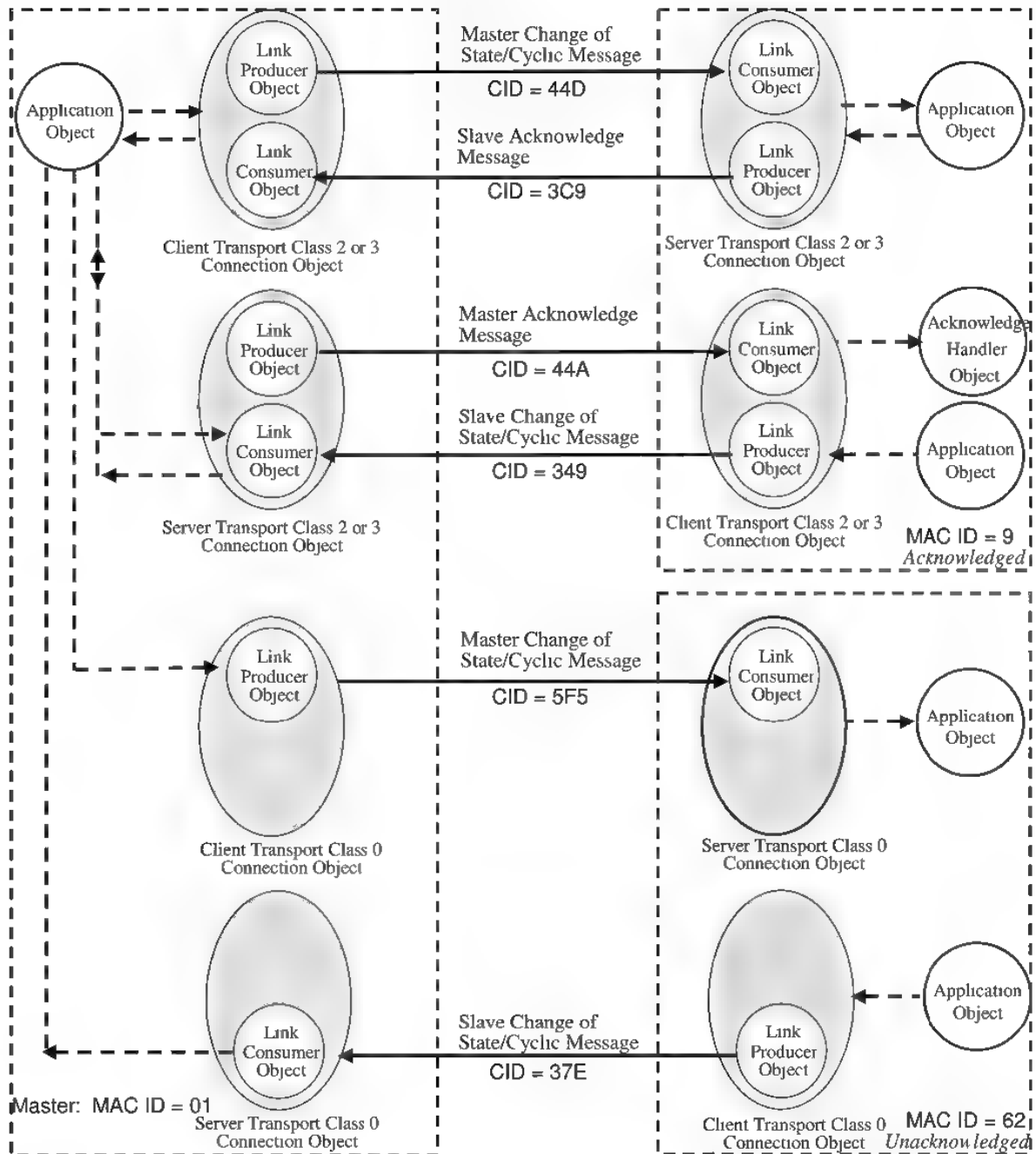


All MAC IDs are shown as decimal numbers.

As shown in Figure 3-13.5, each Change of State/Cyclic message is transmitted on a point-to-point connection from the Master to each Slave. Each Slave’s Change of State/Cyclic message is transmitted back to the Master on separate point-to-point connections. Each connection has its own Connection ID.

Figure 3-13.6, Table 3-13.1 and Table 3-13.2 show how `produced_connection_id` and `consumed_connection_id` attributes (CIDs) for the Change of State/Cyclic Connections between the Master and Slave with MAC ID 9 are determined.

Figure 3-13.5 Example of Change of State/Cyclic Connections



All MAC IDs are shown as decimal numbers  
The Connection ID (CID) is the value in the CAN Identifier Field and is shown in hex.

The figure and tables below show how `produced_connection_id` and `consumed_connection_id` attributes for the Change of State/Cyclic connections within the Master and the Slave whose MAC ID is 9 are determined. Included is a summary of bit values for the Change of State/Cyclic message (Table 3-13.1) and for the Acknowledge message (Table 3-13.2) associated with it.

Figure 3-13.6 Change of State/Cyclic Predefined Master/Slave Connection Identifier Fields

IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE		
10	9	8	7	6	5	4	3	2	1	0				
0	Group 1 Message ID				Source MAC ID				Group 1 Messages			000 – 3ff		
0	1	1	0	1	Source MAC ID				Slave’s I/O Change of State or Cyclic Message			340 - 37F		
0	1	1	1	1	Source MAC ID				Slave’s I/O Poll Response or COS/Cyclic Ack Msg			3c0 - 3ff		
1	0	MAC ID						Group 2 Message ID		Group 2 Messages			400 - 5ff	
1	0	Slave’s MAC ID						0	1	0	Master’s COS or Cyclic Acknowledge Message			402 - 5FA
1	0	Slave’s MAC ID						1	0	1	Master’s I/O Poll/COS/Cyclic Command Message			405 - 5fd

Table 3-13.1 CAN ID Bit Values for COS/Cyclic Message to MAC ID 9

Because:	Bits	Hold the Value:
The Master COS/Cyclic Message is a Group 2 message.	10, 9	10
The Slave's MAC ID is 9.	8 - 3	001001
The Master COS/Cyclic Connection Message ID is 5.	2 - 0	101

A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Master's Change of State/Cyclic Connection in this example, the **produced\_connection\_id** would be set to 44D<sub>hex</sub> which results in the bit pattern 100 0100 1101 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within each Slave's Change of State/Cyclic Connection is set accordingly. See Figure 3-13.5.

Table 3-13.2 CAN ID Bit Values for COS/Cyclic Message to MAC ID 9

Because:	Bits	Hold the Value:
The Slave Acknowledge is a Group 1 message.	10	0
The Slave Acknowledge Message ID is F.	9 - 6	1111
The Slave's MAC ID is 9.	5 - 0	001001

A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Slave's Change of State/Cyclic Connection in this example, the **produced\_connection\_id** would be set to 3C9<sub>hex</sub> which results in the bit pattern 011 1100 1001 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within the Master's Change of State/Cyclic Connection Object for this Slave is set accordingly. See Figure 3-13.5.

Table 3-13.3 Bit Values for Slave COS/Cyclic Message Connection ID from MAC ID 9

Because:	Bits	Hold the Value:
The Slave COS/Cyclic Command is a Group 1 message	10	0
The Slave COS/Cyclic Command Message ID is D.	9 - 6	1101
The Slave's MAC ID is 9.	5 - 0	001001

A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Master's Change of State/Cyclic Connection in this example, the **produced\_connection\_id** would be set to 349<sub>hex</sub> which results in the bit pattern 011 0100 1001 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within each Slave's Change of State/Cyclic Connection is set accordingly. See Figure 3-13.5

**Table 3-13.4 CAN ID Bit Values for Master COS/Cyclic Acknowledge to MAC ID**

Because:	Bits	Hold the Value:
The Master COS/Cyclic Acknowledge is a Group 2 message	<b>10, 9</b>	10
The Slave's MAC ID is 9.	<b>8 – 3</b>	001001
The Master COS/Cyclic Acknowledge Message ID is 2	<b>2 – 0</b>	010

A transmitting Connection Object places its **produced\_connection\_id** attribute within the CAN Identifier Field. With respect to the Slave's Change of State/Cyclic Connection in this example, the **produced\_connection\_id** would be set to 44A<sub>hex</sub> which results in the bit pattern 100 0100 1010 in the CAN Identifier Field. The **consumed\_connection\_id** attribute within the Master's Change of State/Cyclic Connection Object for this Slave is set accordingly. See Figure 3-13.5.

Table 3-13.5 lists the **produced\_connection\_id** attributes associated with both the Master's Change of State/Cyclic Connections and the Slave's Change of State/Cyclic Connections in this example. These are the values that are placed in the CAN Identifier Field when the Connection Objects produce data. Figure 3-13.7 shows non-fragmented example messages.

**Table 3-13.5 MAC ID Conversion for this Example**

Slave MAC ID (decimal)	Slave MAC ID (hex)	Slave MAC ID (binary)	Master COS/Cyclic Connection's produced connection_id attribute (binary)	Master COS/Cyclic Connection's produced connection_id attribute (hex)	Slave COS/Cyclic Acknowledge Connection's produced connection_id attribute (binary)	Slave COS/Cyclic Acknowledge Connection's produced connection_id attribute (hex)
9	09	00 1001	100 0100 1101	44D	011 1100 1001	3C9
62	3E	11 1110	101 1111 0101	5F5	011 1111 1110	3FE
Slave MAC ID (decimal)	Slave MAC ID (hex)	Slave MAC ID (binary)	Slave COS/Cyclic Connection's produced_ connection_id attribute (binary)	Slave COS/Cyclic Connection's produced_ connection_id attribute (hex)	Master COS/Cyclic Acknowledge Connection's produced_ connection_id attribute (binary)	Master COS/Cyclic Acknowledge Connection's produced_ connection_id attribute (hex)
9	09	00 1001	011 0100 1001	349	100 0100 1010	44A
62	3E	11 1110	011 0111 1110	37E	101 1111 0010	5F2

Figure 3-13.7 Non-Fragmented COS/Cyclic Command and Acknowledge Example Messages

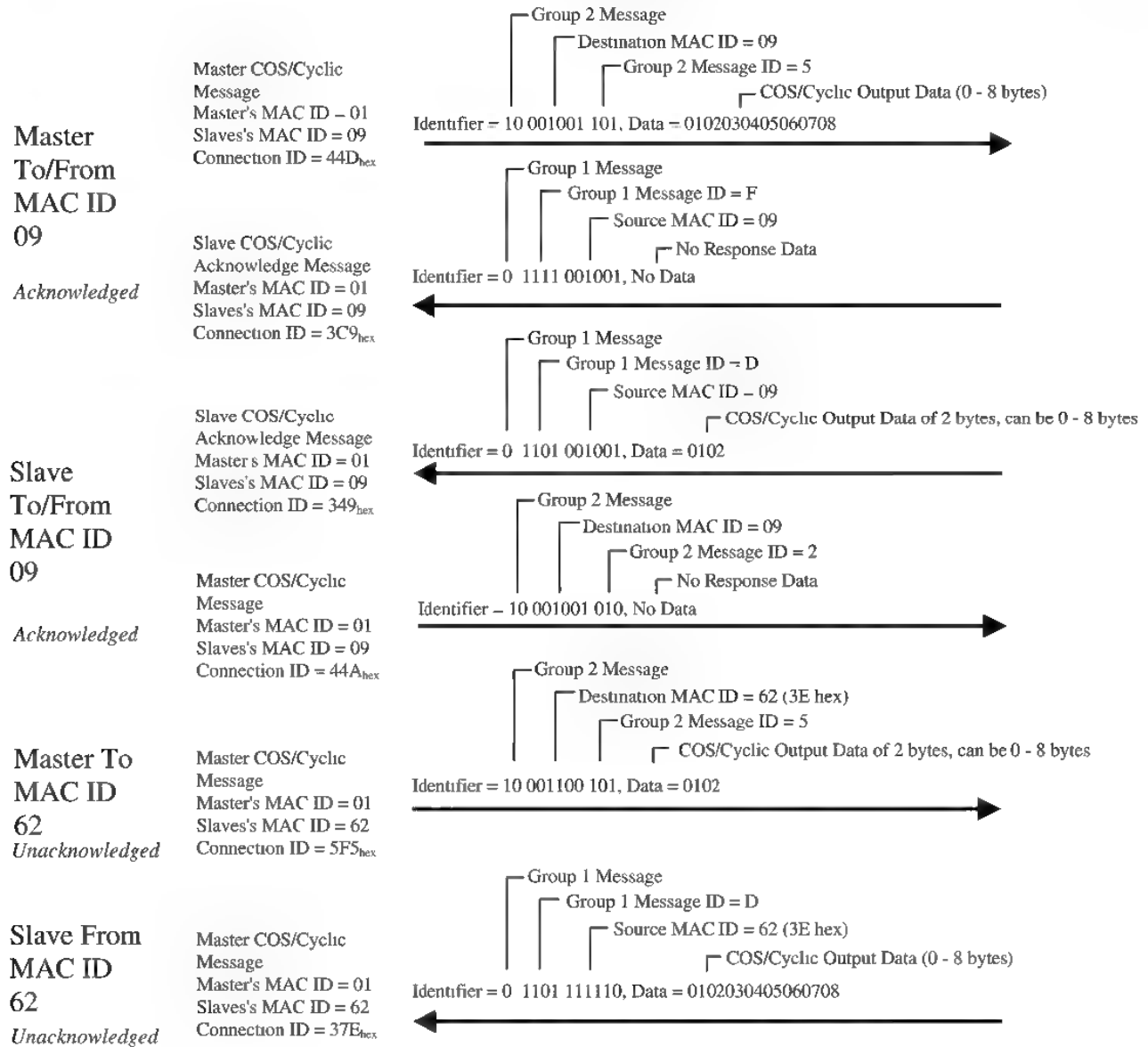


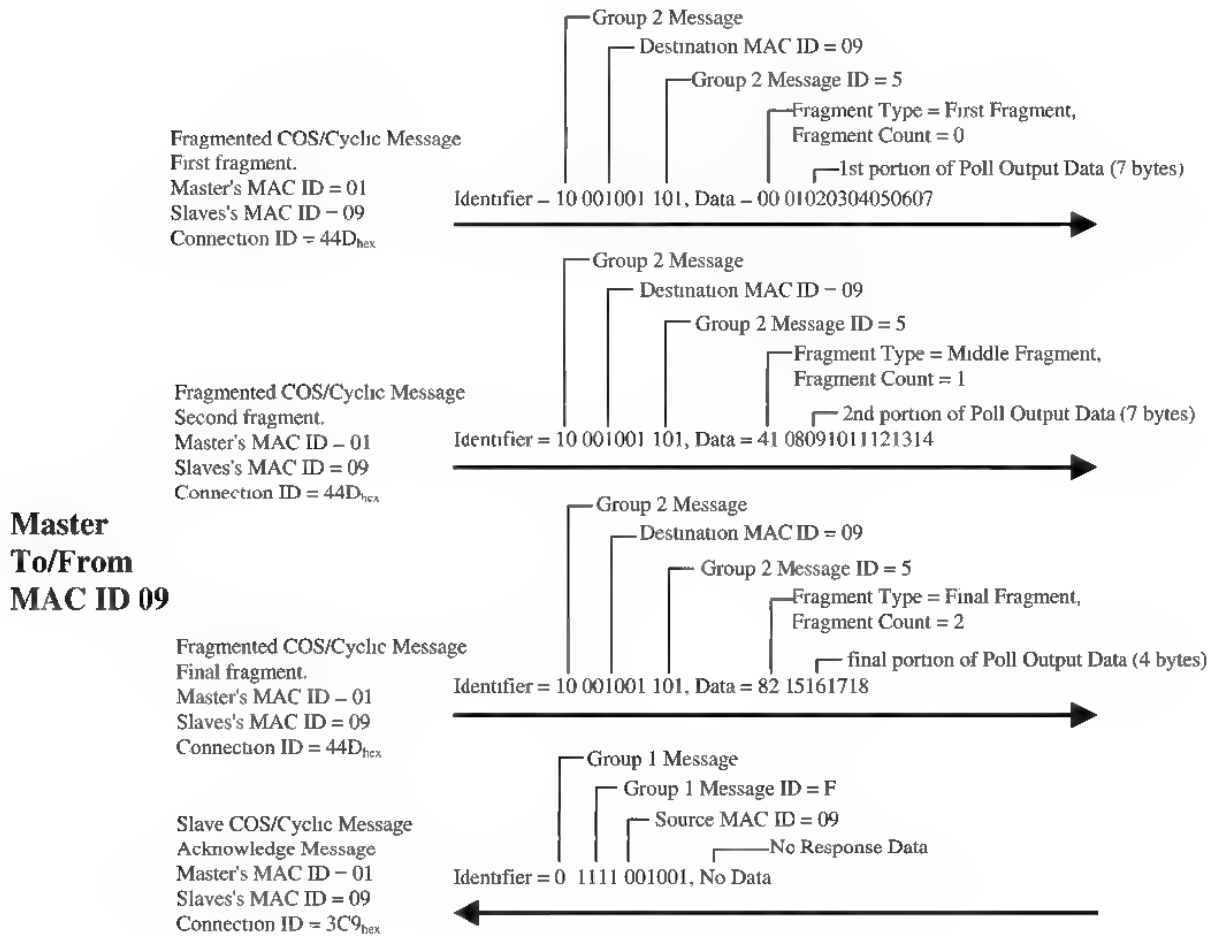


Figure 3-13.8 shows an example of a fragmented Master COS/Cyclic Command message. In this example the Master is sending 18 bytes of output data to slave with MAC ID 9. The Master has set the **connection size** attribute to 12<sub>hex</sub> (18 decimal). The Slave is responding with a non-fragmented COS/Cyclic Acknowledge message.

**Figure 3-13.8 Fragmented COS/Cyclic Command and Acknowledge Example Messages**

Data in COS/Cyclic message is 18 bytes long.

Output data is: 010203040506070809101112131415161718



### 3-14 Group 2 Only Devices

To establish communications with a Group 2 Only device, a Client must allocate the Predefined Master/Slave Connection Set (the Explicit Messaging Connection at a minimum). The request to allocate a Group 2 Only device is transmitted as a Group 2 Only Unconnected Explicit Request. A general overview of this process is presented in this section. Section 3-15 presents a detailed description.

Instead of using the UCMM (in Group 3) to establish Explicit Messaging Connections, Group 2 Only devices (UCMM incapable) receive and process Group 2 Only Unconnected Explicit Request messages (see Figure 3-14.1). Group 2 Only Unconnected Requests are specified by transmitting a Group 2 Message whose MAC ID component contains the intended recipient's MAC ID (Destination MAC ID) and whose Message ID component is set to 6.

**Important:** Rules defining when a Group 2 Unconnected Request message can be transmitted are presented in section 3-15.

**Figure 3-14.1 Predefined Master/Slave Connection Set Identifier Fields**

IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MAC ID						Group 1 Messages	000 – 3ff
0	1	1	0	0	Source MAC ID						Slave's I/O Multicast Poll Response Message	
0	1	1	0	1	Source MAC ID						Slave's I/O Change of State or Cyclic Message	
0	1	1	1	0	Source MAC ID						Slave's I/O Bit Strobe Response Message	
0	1	1	1	1	Source MAC ID						Slave's I/O Poll Response or Change of State/Cyclic Acknowledge Message	
1	0	MAC ID						Group 2 Message ID			Group 2 Messages	400 – 5ff
1	0	Source MAC ID						0	0	0	Master's I/O Bit–Strobe Command Message	
1	0	Multicast MAC ID						0	0	1	Master's I/O Multicast Poll Command Message	
1	0	Destination MAC ID						0	1	0	Master's Change of State or Cyclic Acknowledge Message	
1	0	Source MAC ID						0	1	1	Slave's Explicit Response Messages	
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages	
1	0	Destination MAC ID						1	0	1	Master's I/O Poll/Change of State/Cyclic Message	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages (reserved)	
1	0	Destination MAC ID						1	1	1	Duplicate MAC ID Check Messages	

The only valid Services that can be transmitted as Group 2 Only Unconnected Explicit Request Messages are:

- Allocate\_Master/Slave\_Connection\_Set Message
- Release\_Master/Slave\_Connection\_Set Message

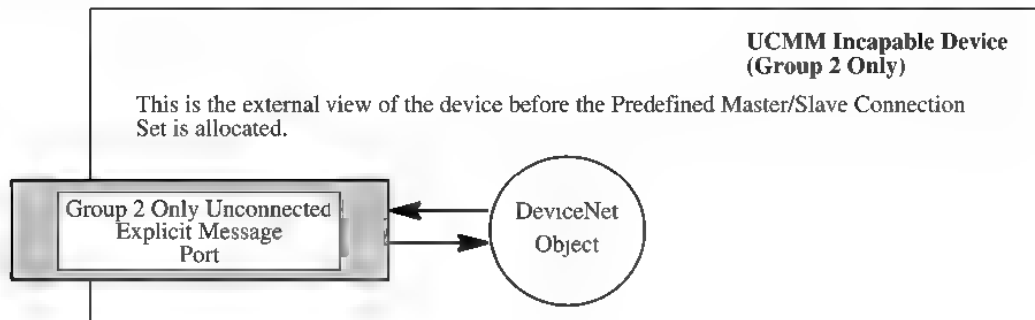
These services are described in detail in section 3-5.

Responses to Group 2 Unconnected Explicit Requests are returned by transmitting a Group 2 Message whose MAC ID component is set to the responding device's MAC ID (Source MAC ID) and whose Message ID is set to 3 (indicated as *Slave's Explicit Response Messages* in Figure 3-14.1). Note that this CAN Identifier has a dual-purpose. It is used to send replies to Group 2 Unconnected Request Messages as well as to reply to requests sent across the Predefined Master/Slave Connection Set's Explicit Messaging Connection.

**Important:** Group 2 Only Devices must reserve the *Slave's Explicit Response Message* Identifier Field for use only with transmitting Group 2 Unconnected response messages and Predefined Master/Slave Explicit Messaging Connection response messages.

After the Group 2 Only device is on-line, but before allocation of the Predefined Master/Slave Connection Set, the Group 2 Only Unconnected Explicit Request Message Port and the Duplicate MAC ID Check Message port are the only ports active. Figure 3-14.2 shows the external view of a Group 2 Only device before the Predefined Master/Slave Connection Set is allocated. Note that although the Duplicate MAC ID Check Message Port is not shown in Figure 3-14.2 all devices are required to support it at all times.

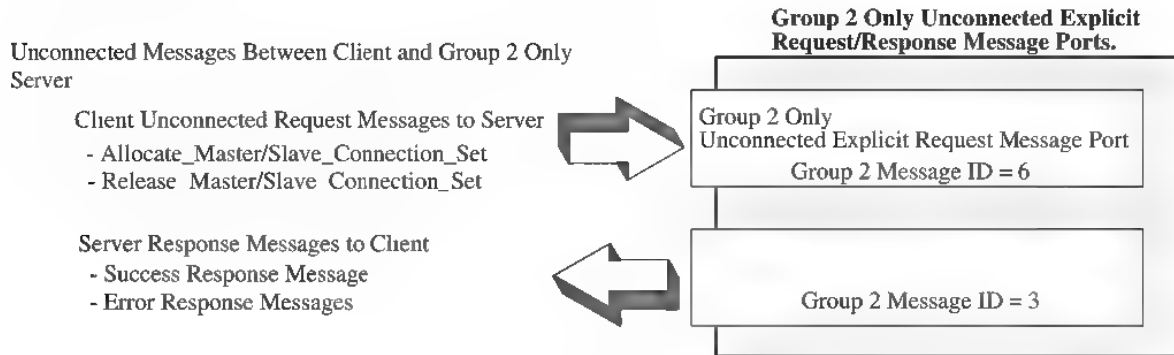
**Figure 3-14.2 Example of Group 2 Only Device BEFORE Allocation of the Predefined Master/Slave Connection Set**



As Figure 3-14.3 indicates, the following messages flow to and from the Unconnected port of a Group 2 Only device:

- Allocate\_Master/Slave\_Connection\_Set Message
- Release\_Master/Slave\_Connection\_Set Message
- Success Response Message to Allocate/Release Master/Slave Connection Set Message
- Error Response Message to Allocate/Release\_Master/Slave\_Connection\_Set Message

**Figure 3-14.3 Message Flow to and from the Group 2 Only Unconnected Explicit Request/Response Message Ports using Predefined Master/Slave Connection Set**

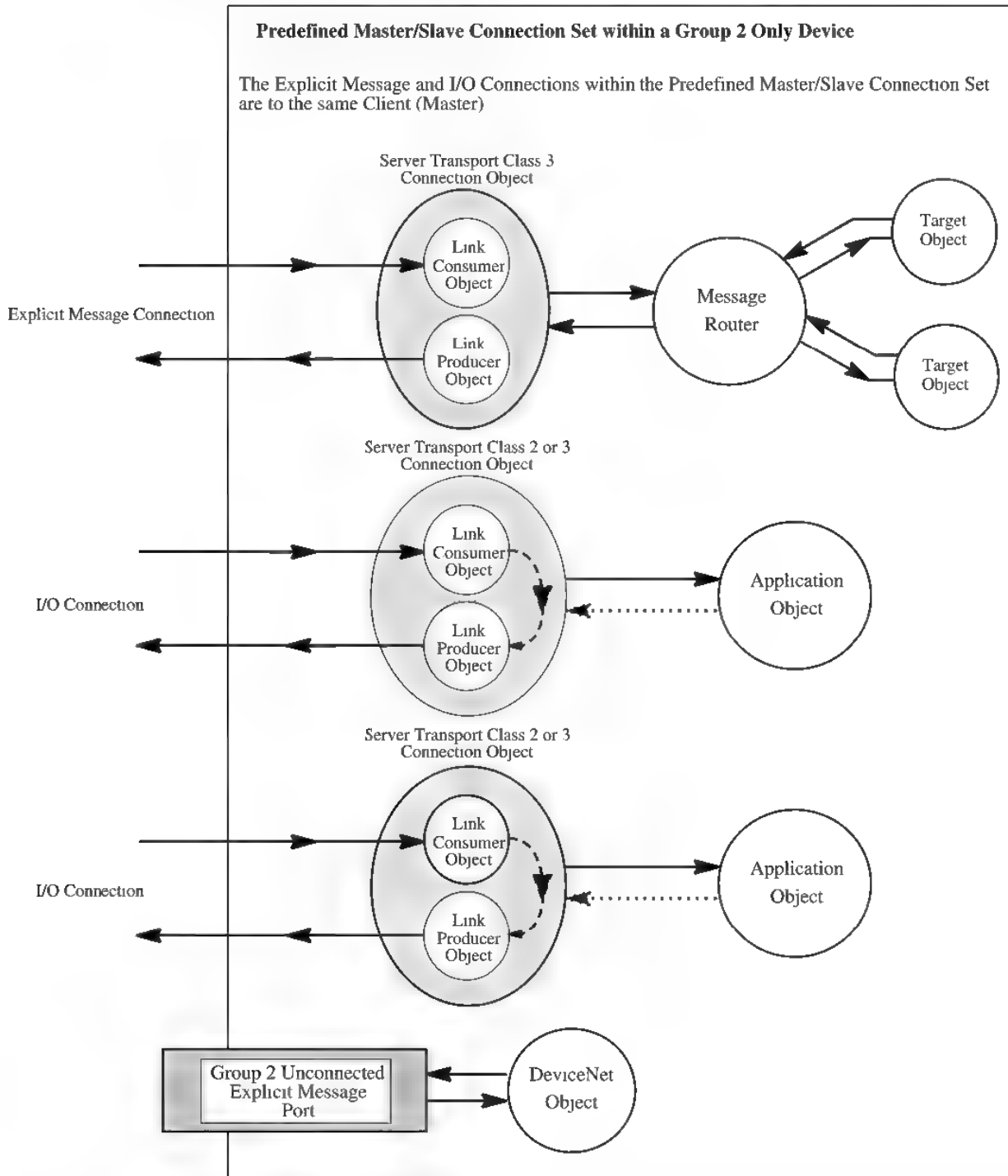


Before the Slave's Explicit Request Messaging Connection is available to receive messages, the Master must allocate it. Once the Master allocates the Master/Slave Explicit Messaging Connection, that Master has unrestricted access to it.

See Figure 3-14.4 for an illustration of a Group 2 Only Device AFTER allocation of the Predefined Master/Slave Connection Set. Note that although the Duplicate MAC ID Check Message Port is not shown in Figure 3-14.4 all devices are required to support it at all times.

If a Group 2 Only Server (UCMM incapable) receives a Group 2 Only Unconnected Request Message that is not an Allocate or Release\_Master/Slave\_Connection\_Set Request, then an Error Response whose Status Code is set to 02 (*Resource Unavailable*), with the Additional Code set to an DeviceNet Object Specific value of 03 is returned (see section 3-5.5).

**Figure 3-14.4 Example - Group 2 Only Device AFTER Allocation of Predefined Master/Slave Connection Set**



### **3-14.1 Limitations of Group 2 Only Devices**

The UCMM is used to establish connections over which all communications with the device occur. On DeviceNet the UCMM is accessed via the Unconnected Explicit Request/Response Message ports within Group 3. Since the Source MAC ID is used in the Identifier field, each access to the UCMM on DeviceNet results in a unique Arbitration field. CAN's bit-wise arbitration ensures that only one transmitter wins the bus. Group 2 Only devices do not support the UCMM. In order to create the Group 2 Only devices, which only have to screen for their MAC ID (Polled device), it is necessary to put the Destination MAC ID in the Message Group 2 MAC ID field. This creates the problems/issues listed below.

The following are issues/limitations concerning Group 2 Only devices:

1. Due to CAN bit-wise arbitration, only ONE device on the link should send messages to the Group 2 Only device at any point in time. Since Destination MAC ID is used in the MAC ID field this is not enforceable electronically. If two devices try to access a Group 2 Only device at the same instant using Destination MAC ID, both will win arbitration on the bus. A bit-error will probably occur due to data mismatch causing both transmitters to retry the message (at the same time again). Both transmitters will continue to retry until either:

- A. One of the messages gets through, at which point the other one may get through, or
- B. One or both go Bus-Off.

If a User accidentally puts the same Group 2 Only device in two Master's scan lists and both masters try to allocate the slave at the same time, this may occur.

2. Since Destination MAC ID is used in the Group 2 Only Server's (Slave's) Explicit Request Message port only the Client can send messages to this port due to number 1. above.

Section 3-15.1, Becoming a Master Using the Predefined Master/Slave Connection Set, describes a procedure that solves or minimizes the problems stated above.

## **3-15 Using the Predefined Master/Slave Connection Set**

This section describes how to use the Predefined Master/Slave Connection Set and the procedure for allocating the Predefined Master/Slave Connection Set within Group 2 Only devices. This section uses the network terms Client and Server, and the application terms Master and Slave. The terms are related as indicated below:

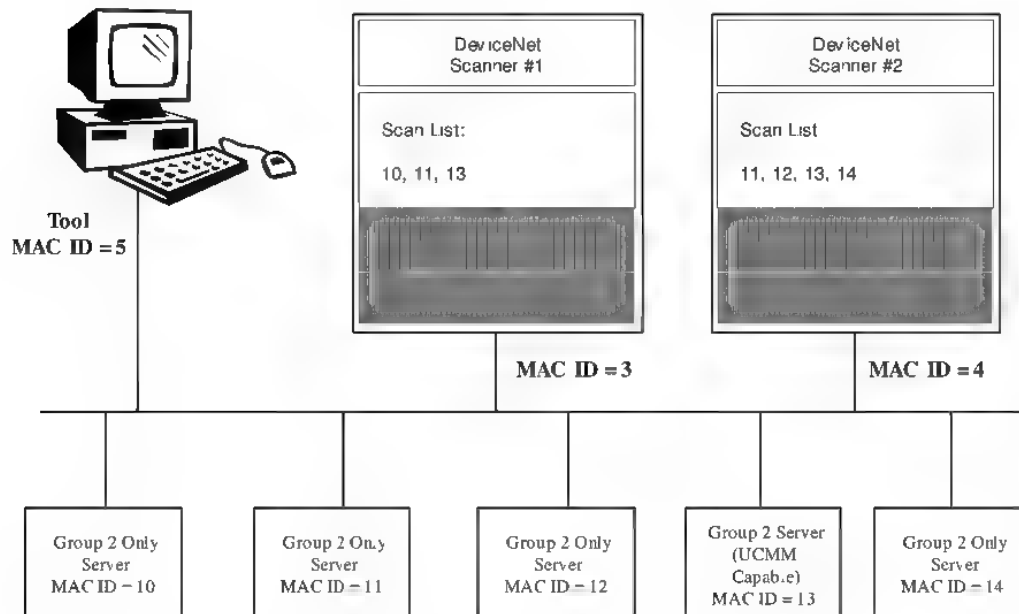
- Client or Group 2 Client = Master
- Server or Group 2 Server = Slave
- Explicit Message Client = the Client side of a Explicit Message connection
- Explicit Message Server = the Server side of a Explicit Message connection
- I/O Client = the Client side of an I/O connection
- I/O Server = the Server side of an I/O connection

The list below presents logic associated with determining whether or not use of the Predefined Master/Slave Connection Set is required to communicate with a device:

- If you have determined that you are attempting to converse with a Group 2 Only Server (two Group 3 UCMM *Open Explicit Message Connection Request* time-outs), then you must allocate the Predefined Master/Slave Connection Set to communicate with the device. At a minimum, this would call for the allocation of the Explicit Messaging Connection associated with the Predefined Master/Slave Connection Set.
- If you want to act as the Client across one or more of the Predefined Master/Slave Connections, then you must allocate and use the desired Connection(s).

Figure 3-15.1 is referenced throughout this section. The Scanners represent DeviceNet Clients (Masters); a Group 2 Client (Master) is the owner of a Group 2 Server.

**Figure 3-15.1 Master/Slave Example Network**



Note that the duplicate Slave MAC IDs in the scan lists above illustrate an important error scenario that is covered in the text that follows.

### 3-15.1 Becoming a Master Using the Predefined Master/Slave Connection Set

A device that wants to function as a Group 2 Client must first allocate the desired Group 2 Connection(s) within the Server. Section 3-5 contains detailed information on allocating the Predefined Master/Slave Connections. Also see Chapter 5 for DeviceNet Object details.

The list below defines the procedure that is followed to allocate the Predefined Master/Slave Connection Set.

#### Predefined Master/Slave Connection Set Allocation Procedure

1. If the Server has been configured as a Quick Connect node within the Client's configuration **and** the Client has detected the presence of the Server by receiving the Server's Duplicate MAC ID message (Client is in Quick Connect mode), the Client transmits both the Open Explicit Message Connection Request to the Server's UCMM and the Allocate\_Master/Slave\_Connection\_Set message to the Server's Group 2 Only Unconnected Explicit Request Message port. Otherwise, the Client determines if the Server is a Group 2 Only Server by transmitting the Open Explicit Message Connection Request to the Server's UCMM.
2. Client starts the *wait\_for\_response* timer. The minimum time-out value for the *wait\_for\_response* timer is one (1) second.
  - If the Server responds successfully (from its UCMM), then the device is UCMM capable. Go to Step 3.
  - If the Server responds successfully from its Group 2 Only Unconnected Explicit Request port (only when Client is in Quick Connect mode), then the Server is allocated to the Client at this point.
  - If the Server does not respond (a *wait\_for\_response* timeout occurs) **and** the Client *is* in Quick Connect mode, then retry transmission of both the Open Explicit Message Connection Request to the Server's UCMM and the Allocate\_Master/Slave\_Connection\_Set message to the Server's Group 2 Only Unconnected Explicit Request Message port. Start the *wait\_for\_response* timer again.
    - If the Server responds successfully from its UCMM, then the device is UCMM capable. Go to Step 3.
    - If the Server responds successfully from its Group 2 Only Unconnected Explicit Request Message port, then the Server is allocated to the Client at this point.
    - If another timeout occurs, then the Client assumes that the Server device is not present on the link.
  - If the Server does not respond (a *wait\_for\_response* timeout occurs) **and** the Client *is not* in Quick Connect mode, then retry the Open Explicit Message Connection Request to the Server's UCMM and start the *wait\_for\_response* timer again.
    - If a response is received, then the device is UCMM capable. Go to Step 3.
    - If a response still is not received (2 *wait\_for\_response* timeouts), then *assume* this device is a Group 2 Only device (not UCMM capable) and go to Step 5.



3. Server is UCMM Capable. The Client attempts to allocate the Predefined Master/Slave Connection Set by transmitting the Allocate\_Master/Slave\_Connection\_Set message, described in section 0, across the just established Explicit Messaging connection. Sending this message instructs the server to act as a Group 2 Server and that this client is its Master (Group 2 Client).

**Important:** If the Server responds successfully from its UCMM, the Client is free to use either the UCMM generated Explicit Message Connection or the Predefined Master/Slave Connection Set Explicit Message Connection (if allocated) in Group 2. However, if the Predefined Master/Slave Connection Set Explicit Message Connection is allocated it must be used per the Parent Explicit Messaging Connection Logic described in Figure 3-5.3. The Server must be designed to handle both. See Figure 3-15.2 below.

**Important:** If the Server responds with an Error to the Allocate\_Master/Slave\_Connection\_Set message, then assume that it does not recognize the Predefined Master/Slave Connection Set, or that it already functions as a Group 2 Server for another Group 2 Client. The error codes in the error response message can be used to determine this. See section 3-5.5, Chapter 2-7.4 and Volume 1, Appendix B for more information about the error response message and the status codes. This situation is illustrated in Figure 3-15.3.

4. If the Server responds successfully to the Allocate\_Master/Slave\_Connection\_Set message then this client is now the Master (the Group 2 Client) for this device, which is UCMM capable. The Server, in turn, allocates the Predefined Master/Slave Connection Set to the MAC ID associated with the Client. The process of allocating the Predefined Master/Slave Connection Set tells the Server who its Master (Client) is, and keeps the Server from allocating the Predefined Master/Slave Connection Set to another Client.

**Important:** Only one Master (Client) can have the Predefined Master/Slave Connection Set allocated at any given time. The entire Predefined Master/Slave Connection Set is allocated.

**Figure 3-15.2 Allocating a UCMM Capable Device across an Explicit Message Connection**

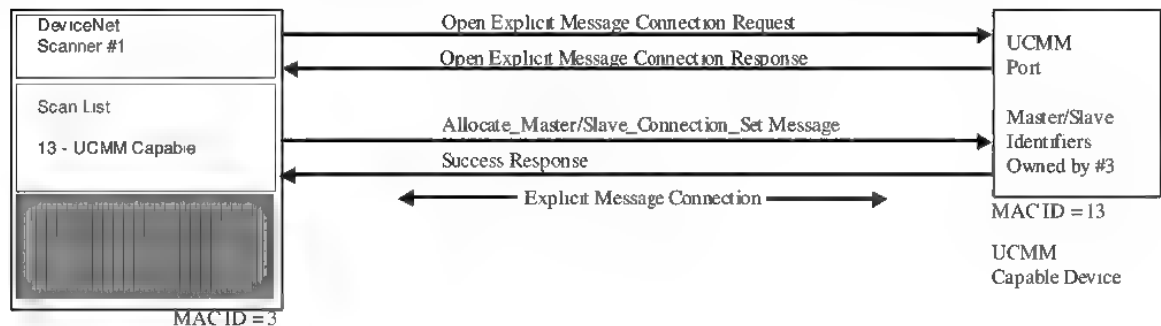
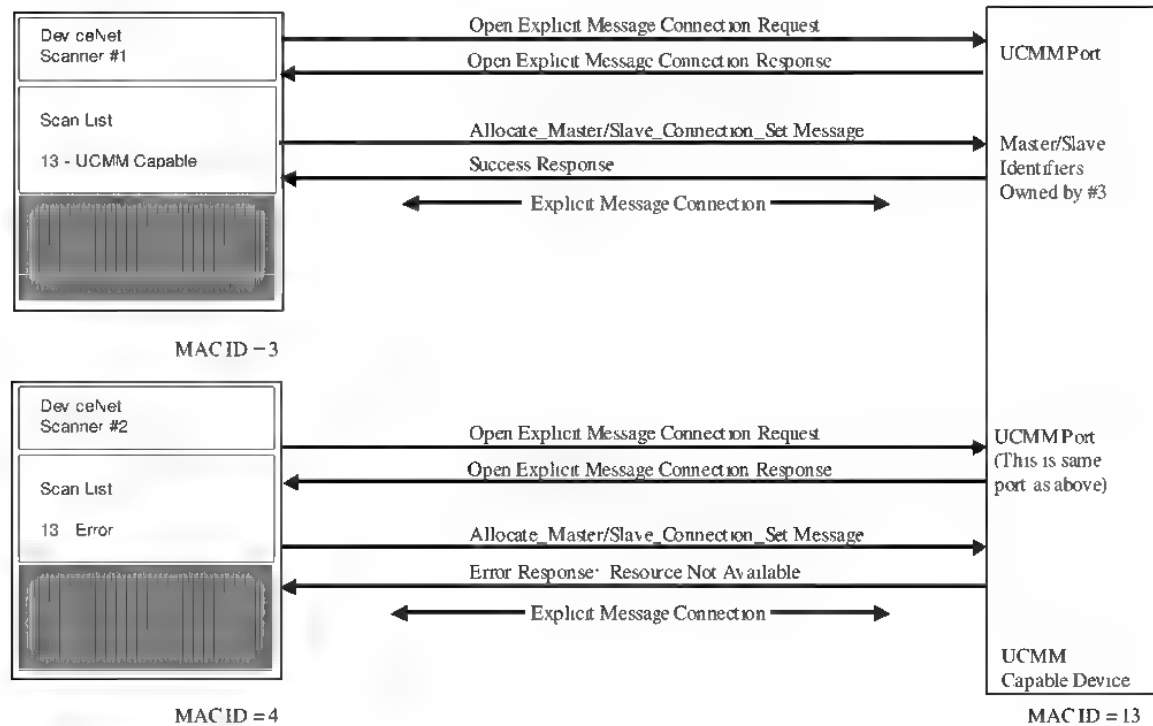


Figure 3-15.3 Server Functions as a Group 2 Server with another Group 2 Client

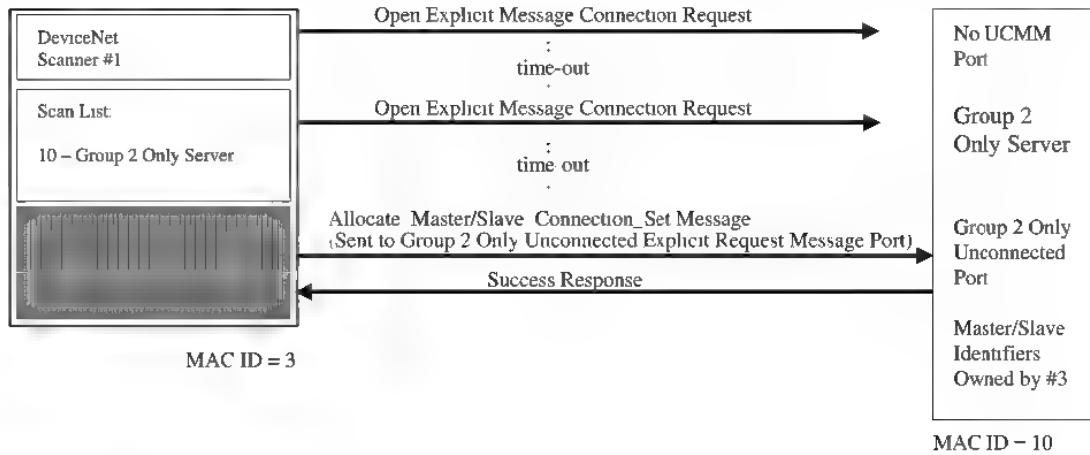


5. Since two Open Explicit Message Connection Request messages, to the Group 3 UCMM of the Server device, have timed-out, the Client assumes that the device is a Group 2 Only server. The Client attempts to allocate the Predefined Master/Slave Connection Set by transmitting the Allocate\_Master/Slave\_Connection\_Set message to the server's Group 2 Only Unconnected Explicit Request Message port as shown in Figure 3-15.4.

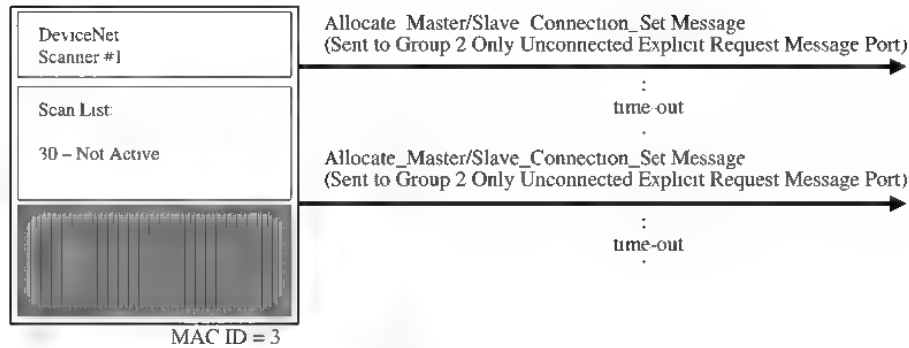
- If the Predefined Master/Slave Connection Set has not already been allocated, the Server responds with a *Success Message* and notes that it has allocated the Connection Set to the MAC ID associated with the Client. See Figure 3-15.4.
- If the Client times-out after transmitting the Allocate\_Master/Slave\_Connection\_Set message to the server's Group 2 Only Unconnected Explicit Request Message port, then the Client sends the message again. If another time-out occurs, then the Client assumes that the Server device is not on the present link. See Figure 3-15.5 .

**Important:** All Group 2 Only Clients must be sure that an Allocate transaction is successful with the Group 2 Only Server before performing any other transactions.

**Figure 3-15.4 Allocation of the Predefined Master/Slave Connection Set by a Group 2 Only Client**



**Figure 3-15.5 Group 2 Server not Present on Link**



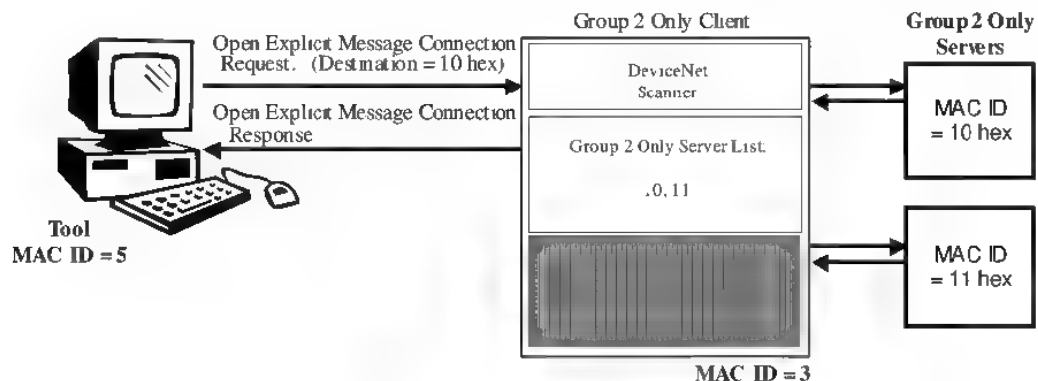
### 3-15.2 Group 2 Only Client Responsibilities

A device that has assumed the responsibility of being a Group 2 Only Client must be capable of doing all of the following:

1. Provide the UCMM for all its Group 2 Only Servers, which means the Group 2 Only Client must:
  - Intercept Unconnected Open Explicit Message Connection Request messages destined for the UCMMs of all its Group 2 Only Servers.
  - Return Open Explicit Message Connection Response messages for all its Group 2 Only Servers. Only a Group Select value of 3 (Message Group 3) will be accepted. A Source Message ID of 4 must be returned in the UCMM open response. Therefore, the Group 2 Only Client will not be able to proxy more than one Explicit Messaging Connection from the same node (configuration tool) to the same Group 2 Only Server (slave). A response message must be generated even if the connection is denied.
2. Always return connection instance 1 in the UCMM Open Response.
3. Intercept Explicit Messaging requests across open connections and send them across the single Group 2 Explicit Message Connection that it owns. The exception is any access to instance 1 of the connection class. In this case, the Group 2 Only Client must perform the function locally and return the appropriate response
4. The Actual Message Body Format field in the Open Explicit Message Connection Response holds the value previously specified by the Group 2 Only Server when the Predefined Master/Slave Identifier Set was allocated.

The process of intercepting and responding to Unconnected Open Explicit Message Connection Requests directed towards a Group 2 Only Server is illustrated below in Figure 3-15.6. Note that the Tool shown in Figure 3-15.6 is just an example of a Client.

**Figure 3-15.6 Interception of and Response to Open Explicit Message Connection Request**



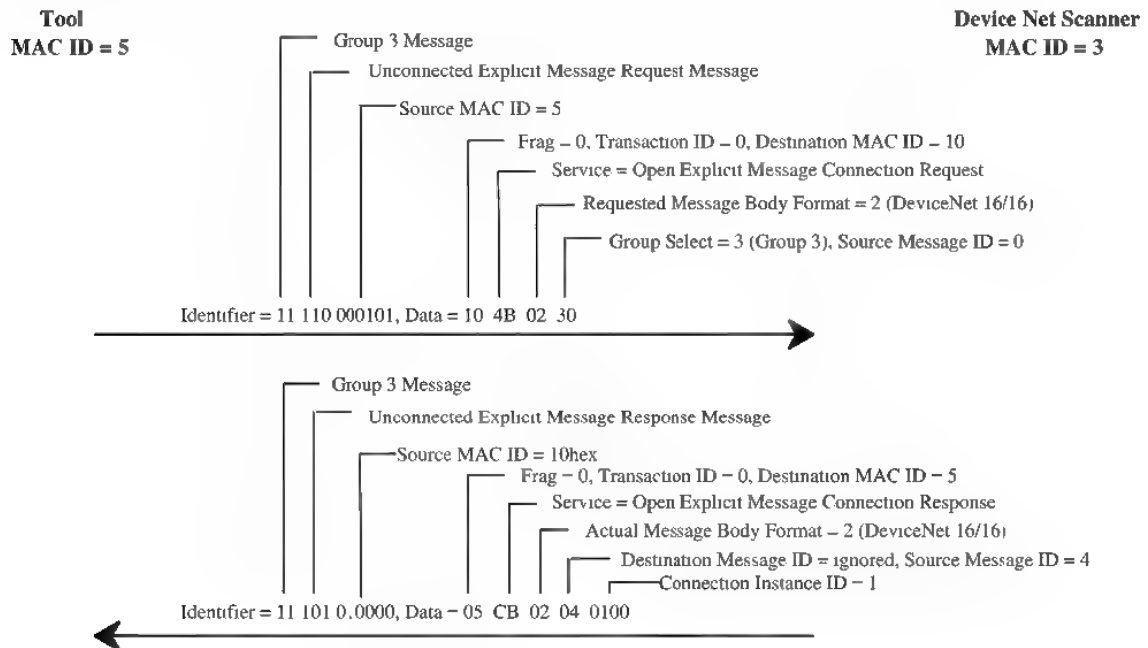
In Figure 3-15.6, the Tool is trying to open a connection to the device with MAC ID 10 hex. It sends an unconnected Open Explicit Message Connection Request message to the UCMM of the server with MAC ID equal to 10 hex. The Scanner (Group 2 Only Client) intercepts this message and responds with an Open Explicit Message Connection Response message. The Tool has no knowledge that the DeviceNet Scanner intercepted its request and responded for the Group 2 Only Server. The Tool assumes it has opened an Explicit Message Connection directly to the Server with MAC ID of 10.

**Important:** The Group 2 Only Client must not allow the Tool to transmit Identifiers already in use by itself and/or the Group 2 Only Server.

From this point forward, the Group 2 Only Client (DeviceNet Scanner in this example) must be prepared to screen for and receive messages transmitted by the Tool across this connection. When a message is received the Group 2 Only Client “forwards” the message across the Predefined Group 2 Master’s Explicit Request Message port to the Group 2 Only Server.

Notice in the example below that the DeviceNet Scanner places the Group 2 Only Server’s MAC ID in the Source MAC ID field of the Identifier when responding to the request. This is permitted because the DeviceNet Scanner knows that MAC ID 10 will never transmit this Identifier. Figure 3-15.7 below illustrates an example of the Open Explicit Message Connection Request/Response exchange described above.

**Figure 3-15.7 Tool Opening Connection with Group 2 Only Server through Client**



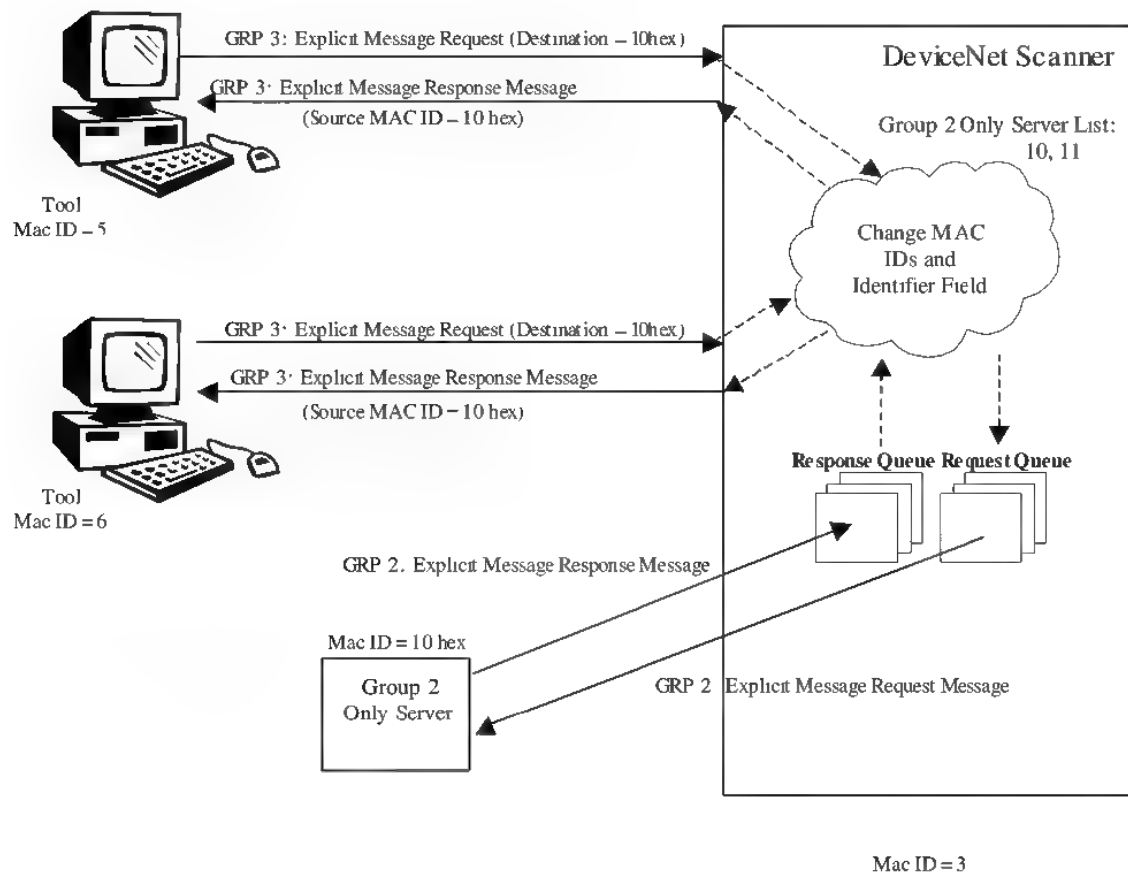
### 3-15.3 Forwarding the Message

The Group 2 Only Client “forwarding” the message associated with taking a request sent from the Tool (example shown in Figure 3-15.8) and generating a Group 2 Only Server Explicit Request Message is characterized below:

- The Group 2 Only Client replaces the MAC ID of the Request Message from the Tool with its own MAC ID. The MAC ID is located within the Message Header of the data field.
- The Identifier Field that the Tool specified in the Explicit Message Request is replaced by the Predefined Group 2 Master’s Explicit Request Identifier Field.

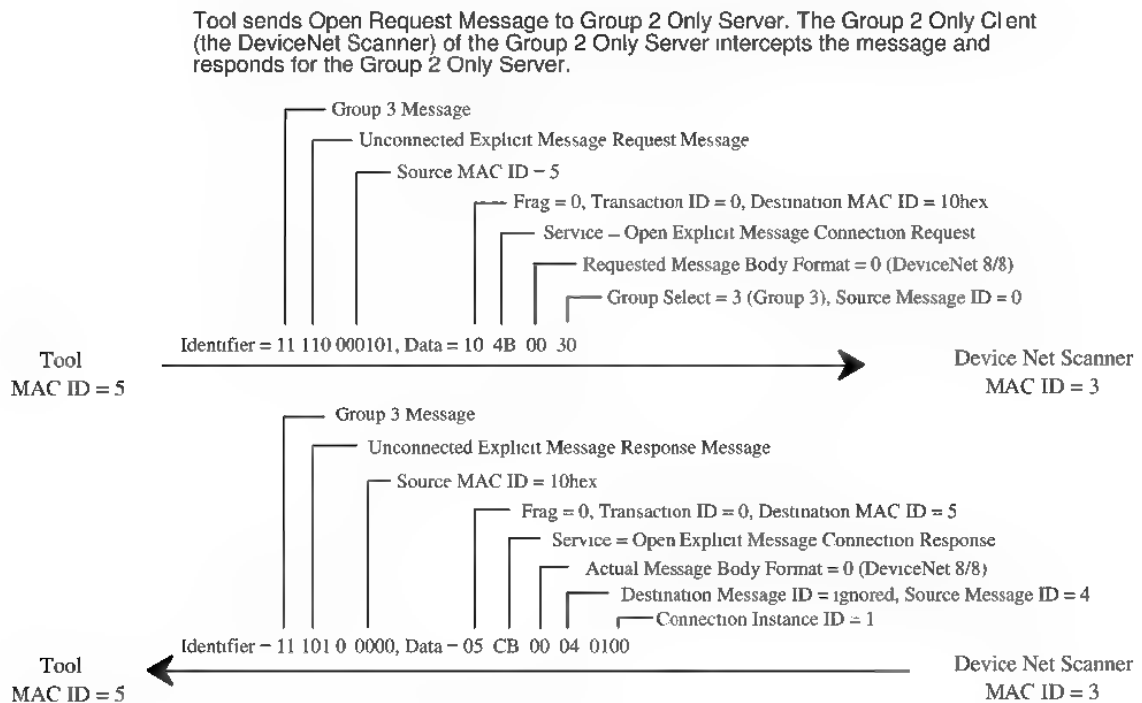
This process is reversed for the response message.

Figure 3-15.8 Forwarding the Message



Using Figure 3-15.8 as an example, Figure 3-15.9 below illustrates the Open Explicit Message Connection Request/Response exchange between the Tool (MAC ID = 5) and a DeviceNet Scanner (MAC ID = 3). The tool is attempting to open an Explicit Messaging connection to a device with MAC ID = 10<sub>hex</sub> by sending an Unconnected Explicit Request Message to the UCMM of the device. However, DeviceNet Scanner with MAC ID = 3 is the Group 2 Only Client of the device and it intercepts and responds to the Unconnected Explicit Request Message, as is shown in Figure 3-15.9. The tool believes it is communicating with a UCMM capable server across an Explicit Message connection. The DeviceNet Scanner is intercepting and forwarding the messages, passing them back and forth between the tool and the Group 2 Only Server.

**Figure 3-15.9 Tool Opening Connection with Group 2 Only Server through Client**



In this example, now that the Tool has a connection it performs a Get\_Attribute\_Single of the Expected Packet Rate (EPR) on the Group 2 Only Server's Connection Instance #2. Figure 3-15.10 shows the DeviceNet Scanner intercepting the Get\_Attribute\_Single EPR Explicit Request Message and forwarding it to the Group 2 Server.

**Figure 3-15.10 Tool Sends Get\_Attribute\_Single in Group 2 Only Server through Client**

The Tool (MAC ID 5) sends a Get\_Attribute\_Single EPR Explicit Request message across the Group 3 Connection to what it thinks is the server (MAC ID 10 hex). The Group 2 Only Client (DeviceNet Scanner with MAC ID 3) of the Group 2 Only Server intercepts the message and forwards it to the Group 2 Only Server (MAC ID 10 hex).

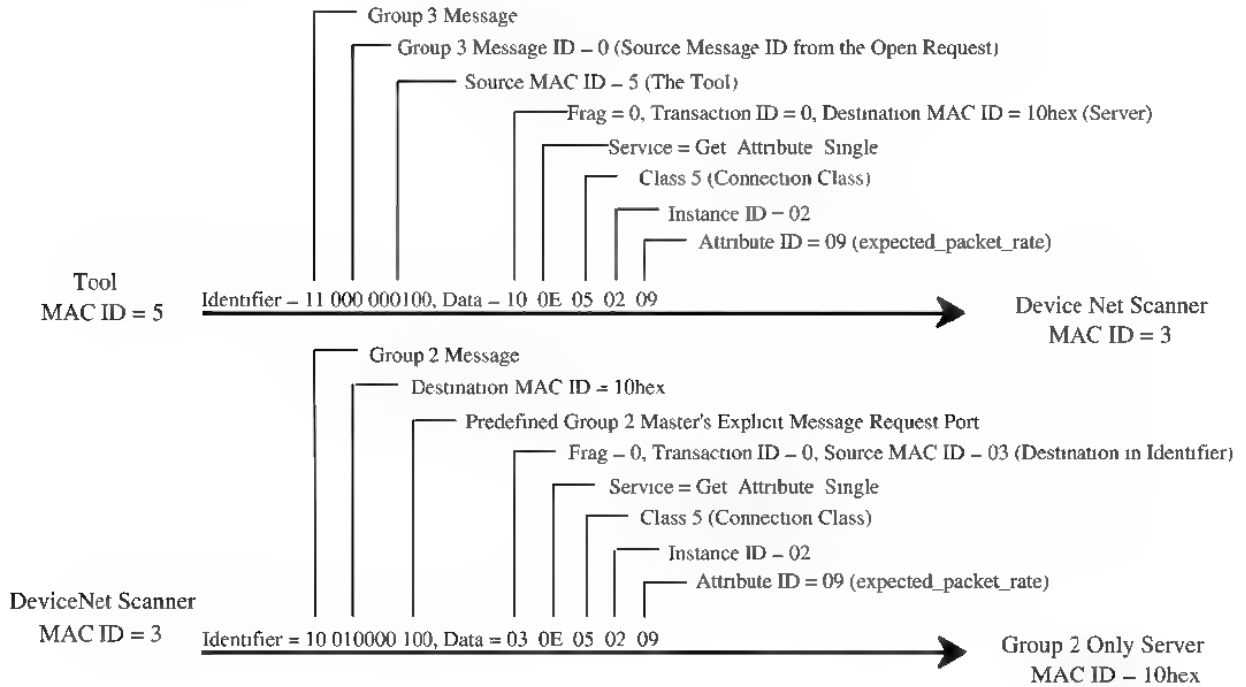
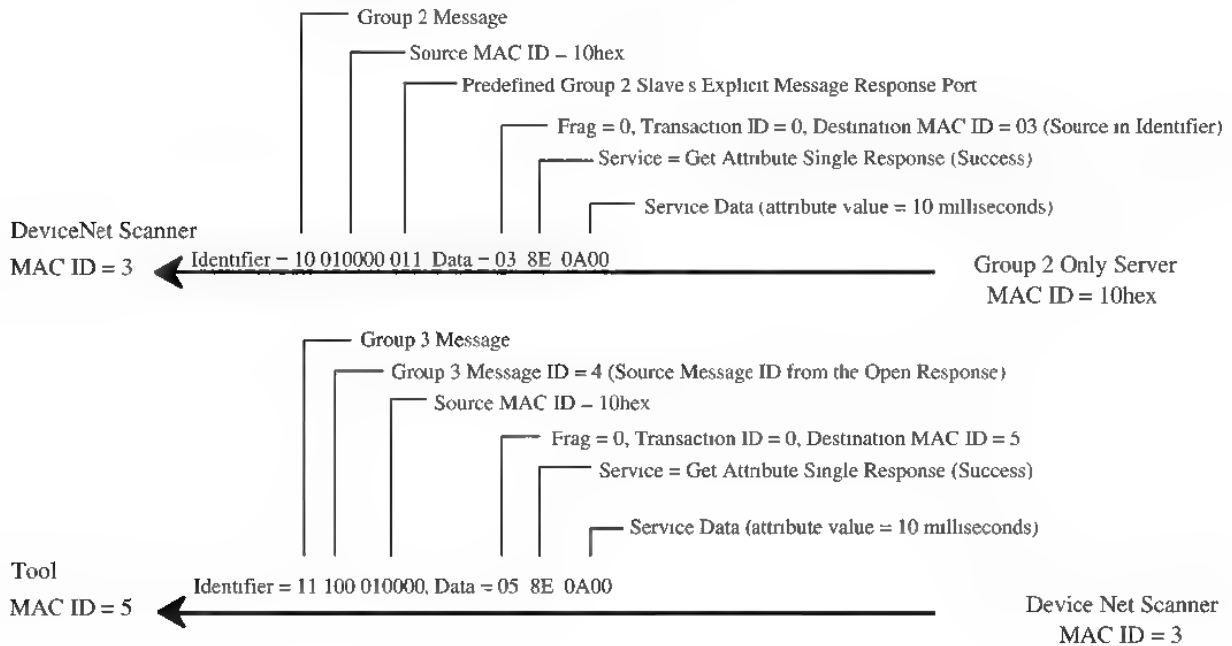




Figure 3-15.11 illustrates the response sequence for the Get\_Attribute\_Single EPR Explicit Message. The DeviceNet Scanner intercepting the Get\_Attribute\_Single EPR Response Message from the Group 2 Only Server and forwarding it to the Tool is shown.

**Figure 3-15.11 Tool Receives Response to Get\_Attribute\_Single EPR Attribute from Group 2 Only Server via Client**

The Group 2 Only Server responds to the Get\_Attribute\_Single EPR Explicit Request message across the Group 2 Connection. The Group 2 Only Client of the Group 2 Only Server (DeviceNet Scanner) forwards the message to the Tool over the Group 3 connection.



### 3-15.4 Possible Group 2 Only Error Scenarios

This section describes errors associated with the Group 2 Only Client that may occur on the DeviceNet network. Included is an explanation of the following:

- Loss of Group 2 Only Client
- Duplicate Attempts to become a Group 2 Only Client

#### 3-15.4.1 Loss of Group 2 Only Client

If the Group 2 Only Client should fault or otherwise stop communication on DeviceNet, the network will experience the following:

- All connections time-out and are either closed or put into the timed-out state by the Group 2 Only Server.
- The Group 2 Only Server releases (deallocates) the Predefined Master/Slave Connection Set at which time any other Client is free to gain ownership of the Server.
- All connections used by a tool communicating with the Group 2 Only Server, through the Group 2 Only Client, time-out. The tool may think that the Group 2 Only Server left the link.

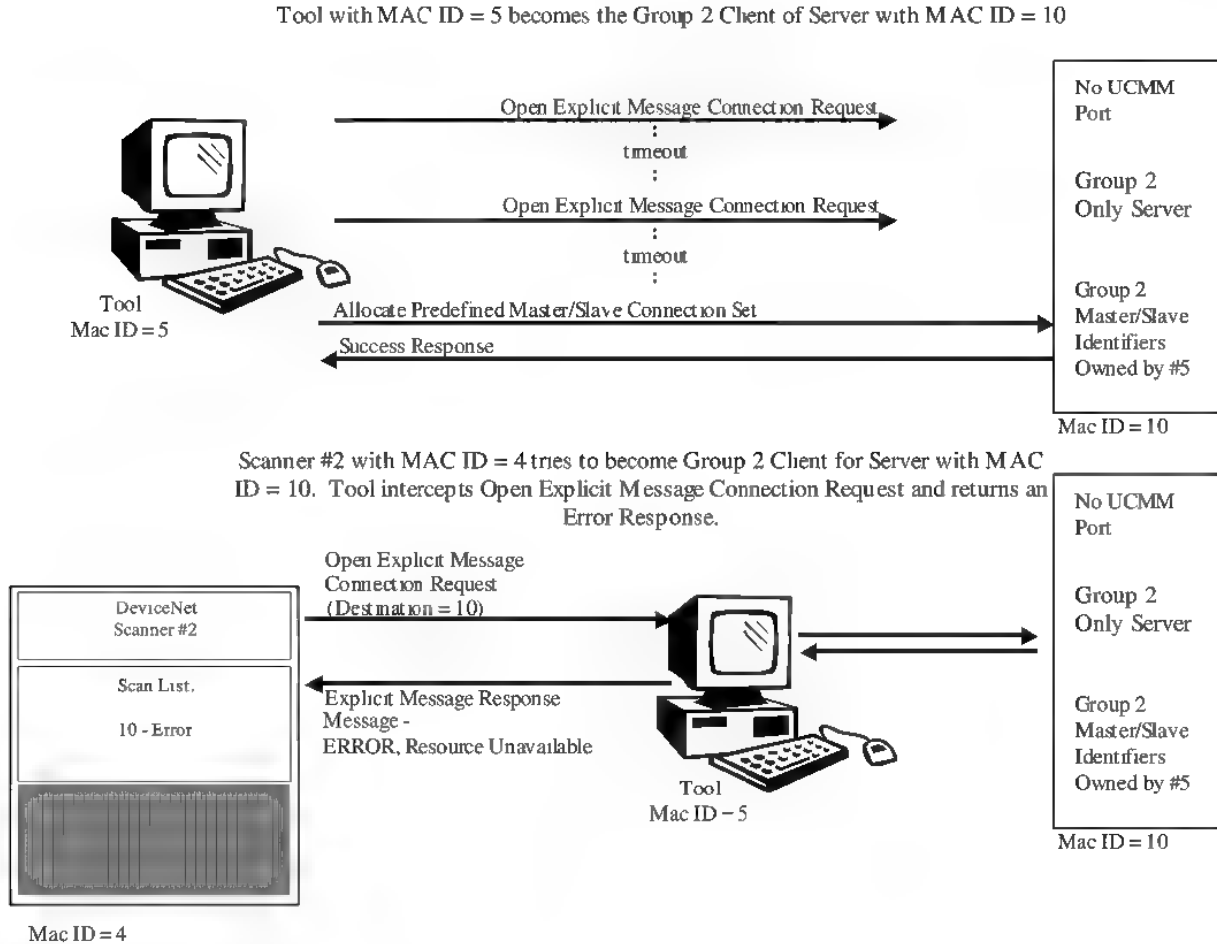
If a tool has communications in progress with the Group 2 Only Server when the Group 2 Only Client stops communicating, the connection the tool is using times-out and is closed. The tool may try to reconnect and become the Group 2 Only Client, as is shown in the upper half of Figure 3-15.12. Note that the tool must try to establish communication with the Group 2 Only Server as any Client does; try the UCMM twice and then the Group 2 Unconnected Port.

If another Client tries to connect with the Group 2 Only Server at about the same time the Tool does, a duplicate attempt is detected and the Duplicate Attempt Detection algorithm followed. See section 3-15.4.2 for details concerning *Duplicate Attempts To Become Group 2 Only Client*.

Once the Group 2 Only Server releases the Predefined Master/Slave Connection Set any other Client is free to gain ownership of the Server. However, any device that does so must fulfill all of the Group 2 Only Client Responsibilities. If a tool wants to become the Group 2 Only Client of a Group 2 Only Server it must be prepared to do this. However, the tool can reject all unconnected Open Explicit Message Connection Request messages heading to its Group 2 Only Server by responding with an Error message. This is shown in the lower half of Figure 3-15.12.

**Important:** Any device which is the Group 2 Only Client of a Group 2 Only Server must fulfill all of the Group 2 Only Client Responsibilities.

Figure 3-15.12 Tool is Group 2 Only Client



### 3-15.4.2 Duplicate Attempts to Become Group 2 Only Client

When two devices attempt to become Group 2 Only Clients at the *same time* to the *same* Group 2 Only Server, both attempt to open an Explicit Messaging Connection, both time-out twice, and both attempt to allocate the Predefined Master/Slave Connection Set by transmitting the Allocate\_Master/Slave\_Connection\_Set request message. The possible transmission of the Allocate\_Master/Slave\_Connection\_Set request message by both Clients at the same time can result in the situation detailed in section 3-14.1, Limitations of Group 2 Only Devices.

**Important:** To minimize the chance of two devices trying to become Group 2 Only Clients of the same Group 2 Only Server at the same time, any Client attempting to become a Group 2 Only Client must listen to all Open Explicit Message Connection requests being sent to that server's UCMM and follow the procedure detailed below.

While sending two Open Explicit Message Connection Request messages, Client #1 listens to see if another Client is sending the same messages to the same server. If another Client is sending an Open Explicit Message Connection Request message to the same server AND its MAC ID is lower, then Client #1 must back off and let the other Client proceed. No additional code is required because the Group 2 Only Client must support receiving Open Explicit Message Connection Requests for its Group 2 Only Servers once it has gained ownership.

The following code fragment will help clarify this procedure.

```
serverDevice = nextScanListEntry;
scanList[serverDevice].status = SUCCESS;
retryCount = 0;
sendOpenExplicitMessageConnection(serverDevice);
startTimer();
switch(whatHappensNext)
    case OPEN_REQUEST_TIMEOUT:
        if(retryCount == 0)
            retryCount++;
            sendOpenExplicitMessageConnection(serverDevice);
            startTimer();
        else if(scanList[serverDevice].status != DUPLICATE_ATTEMPT_DETECTED)
            scanList[serverDevice].type = GROUP2_ONLY_SERVER;
            sendAllocateMasterSlaveConnectionSet(across Group 2 Unconnected port);
            startTimer();
        else
            scanList[serverDevice].status = CANT_GAIN_OWNERSHIP;
            break;

    case OPEN_EXPLICIT_MESSAGE_REQUEST_RECEIVED:
        if(destination == myMacId)
            OpenRequestReceived()
        else if((destination == serverDevice) &&
            (sourceOfOpenReq.MacId < myMacId))
            scanList[serverDevice].status = DUPLICATE_ATTEMPT_DETECTED;
        else if(destination == one of my Group 2 Only Servers)
            provideGroup2OnlyClientResponsibilityFunctions();
            break;

    case OPEN_EXPLICIT_MESSAGE_RESPONSE_RECEIVED:
        cancelTimer();
        scanList[serverDevice].type = UCMM_CAPABLE;
        sendAllocateMasterSlaveConnectionSet(across the Explicit Messaging Connection
        we just opened);
        startTimer();
        break;
```

## 3-16 Bit-Strobe and Poll filter Requirements

Because a Master device receives different messages than does a Slave device, each has separate message filtering requirements. The following sections include general information about Bit-Strobe and Poll message filtering. Additional message filtering must be used for Change of State/Cyclic connections.

### 3-16.1 Master Device Filtering

Most likely, the Master device is sending Bit-Strobe and Poll Command messages and wants to receive Bit-Strobe and Poll response messages. In addition the Master is probably sending and receiving Explicit messages. Note that these messages may be to/from both UCMM capable and UCMM incapable devices. To perform its responsibilities, the Master may require identifier filtering for the following messages:

- *Slave's I/O Bit-Strobe Response:* Master watches for Bit-Strobe Response messages from each of its Slaves, as configured.
- *Slave's I/O Poll Response:* Master watches for Poll Response messages from each of its Slaves, as configured.
- *Group 2 Slave's Explicit Response:* For each of its Group 2 Only (UCMM incapable) Slaves, the Master watches for Explicit Response messages using the Predefined Master/Slave Connection Set. For each of its Group 2 (UCMM capable) Slaves the Master uses an Explicit message connection (probably Group 3, not shown in Figure 3-15.9).
- *Duplicate MAC ID Check:* A Master, as do all devices, monitors for Duplicate MAC ID Check messages that match its MAC ID.
- *Group 3 Unconnected Explicit Request/Response:* A Master must be UCMM capable, and it must receive all Explicit Request/Response messages to determine which are meant for it and its Group 2 Only Slaves.

Figure 3-16.1 Master Filtering of Bit-Strobe and Poll Response Messages

IDENTIFIER BITS										IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1		
0	1	1	1	0	Slave's MAC ID					Slave's I/O Bit-Strobe Response Message	380 - 3bf
0	1	1	1	1	Slave's MAC ID					Slave's I/O Poll Response Message	3c0 - 3ff
1	0	Slave's MAC ID					0	1	1	Slave's Explicit Response Messages	403 - 5fb
1	0	Master's MAC ID					1	1	1	Duplicate MAC ID Check Message	407 - 5ff
1	1	1	0	1	Source MAC ID					Unconnected Explicit Response Messages	740 - 77f
1	1	1	1	0	Source MAC ID					Unconnected Explicit Request Messages	780 - 7df

Figure 3-16.1 shows the possible identifier bit patterns that a Master may have to receive. Given the large number of combinations and limited number of CAN acceptance filters, many Masters will need to receive all messages and screen them in software.

### 3-16.2 Slave Device Filtering (Group 2 Only): Bit-Strobe

To perform its duties as a Bit-Strobed Group 2 Only Slave device, the Slave may require identifier filtering for the following messages:

- *Master's I/O Bit-Strobe Command*: The Slave must receive the Bit-Strobe Command message from its Master.
- *Master's Explicit Request*: The Slave must be able to receive Explicit Request messages across the Master's Explicit Request message port.
- *Group 2 Only Unconnected Explicit Request*: The Slave must be able to receive Group 2 Unconnected Explicit Request messages across the Group 2 Only Unconnected Explicit Request message port.
- *Duplicate MAC ID Check*: The Slave must receive Duplicate MAC ID Check messages that match its MAC ID.

A Group 2 Only slave device that wants to accept the Bit Strobe Command message may need to screen anywhere from some to all Group 2 messages in software. This is because the Bit-Strobe command uses Source MAC ID, requiring the Slave to receive messages using its Master's MAC ID and its own MAC ID. See Figure 3-16.2. If the CAN chip being used has enough acceptance filters, the slave can screen in hardware. If the CAN chip has just one Mask-and-Match filter, then the device may filter on Group 2 messages and screen specific messages in software.

If the Slave is filtering Group 2 messages in software, it must receive all Group 2 messages. Bits that are common to both the Master's MAC ID and the Slave's MAC ID further increase what can be filtered.

**Figure 3-16.2 Group 2 Only Slave Filtering - Bit-Strobe**

IDENTIFIER BITS										IDENTITY		HEX RANGE
10	9	8	7	6	5	4	3	2	1	0	USAGE	
1	0	Master's MAC ID						0	0	0	Master's I/O Bit -Strobe Command Message	400 - 5f8
1	0	Multicast MAC ID						0	0	1	Master's I/O Multicast Poll Command Message	401 - 5f9
1	0	Slave's MAC ID						0	1	0	Master's Change of State or Cyclic Acknowledge Message	402 - 5fa
1	0	Slave's MAC ID						0	1	1	Slave's Explicit Response Messages	403 - 5fb
1	0	Slave's MAC ID						1	0	0	Master's Explicit Request Messages	404 - 5fc
1	0	Slave's MAC ID						1	0	1	Master's I/O Poll Command Message	405 - 5fd
1	0	Slave's MAC ID						1	1	0	Group 2 Unconnected Explicit Request Messages	406 - 5fe
1	0	Slave's MAC ID						1	1	1	Duplicate ID Check Message	407 - 5ff

Devices that require a lower interrupt rate can accept the Poll Command message instead of the Bit-Strobe (because the Poll uses the Destination MAC ID, meaning the device only accepts messages destined for it, it receives fewer messages, resulting in fewer interrupts). For more information on the CAN Interrupt Rate, see Chapter 9-12.

### 3-16.3 Slave Device Filtering (Group 2 Only) - Poll

To perform its duties as a Polled Group 2 Only Slave device, the Slave may require identifier filtering for the following messages:

- *Master's I/O Poll Command*: The Slave must receive the Poll Command message from its Master.
- *Master's Explicit Request*: The Slave must be able to receive Explicit Request messages across the Master's Explicit Request message port.
- *Group 2 Only Unconnected Explicit Request*: The Slave must be able to receive Group 2 Unconnected Explicit Request messages across the Group 2 Only Unconnected Explicit Request message port.
- *Duplicate MAC ID Check*: The Slave must receive Duplicate MAC ID Check messages that match its MAC IDs.

A Group 2 Only Polled slave device receives only messages addressed to it. The Poll Command message, the Explicit Request messages, the Group 2 Unconnected Explicit Request messages and the Duplicate MAC ID Check messages all contain the Slave's MAC ID.

By receiving messages destined only for it, the Polled Group 2 Only slave device receives fewer messages, resulting in fewer interrupts.

**Table 3-16.1 Slave Filtering of Poll Commands**

IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1	0		
1	0	Slave's MAC ID						1	0	0	Master's Explicit Request Message	404 5fc
1	0	Slave's MAC ID						1	0	1	Master's I/O Poll Command Message	405 – 5fd
1	0	Slave's MAC ID						1	1	0	Unconnected Explicit Request Messages	406 5fe
1	0	Slave's MAC ID						1	1	1	Duplicate MAC ID Check Message	407 – 5ff

## **Volume 3: DeviceNet Adaptation of CIP**

# **Chapter 4: CIP Object Model**

---



## **Contents**

4-1	Introduction.....	3
-----	-------------------	---

## **4-1 Introduction**

This chapter of the DeviceNet application contains additions to the CIP object model that are DeviceNet specific. At this time, no such additions exist.

This page left intentionally blank

## **Volume 3: DeviceNet Adaptation of CIP**

### **Chapter 5: Object Library**

---

## Contents

5-1	Introduction.....	3
5-2	Reserved Class Codes .....	3
5-3	Identity Object .....	4
5-4	DeviceNet Object Class Definition.. .. .	5
5-4.1	DeviceNet Object Class Attributes .....	5
5-4.1.1	Revision - UINT data type .....	6
5-4.2	DeviceNet Object Class Services .....	6
5-4.3	DeviceNet Object Instance Attributes.....	6
5-4.3.1	MAC ID .....	9
5-4.3.2	Baud Rate.....	10
5-4.3.3	BOI (Bus-Off Interrupt) .....	11
5-4.3.4	Bus off Counter.....	12
5-4.3.5	Allocation Information.....	12
5-4.3.5.1	Allocation Choice Byte.....	12
5-4.3.5.2	Master's MAC ID .....	13
5-4.3.6	Diagnostic Counters.....	13
5-4.3.7	Active Node Table .....	14
5-4.4	DeviceNet Object Instance Services .....	14
5-4.4.1	Common Services .....	14
5-4.4.1.1	Reset Service.....	15
5-4.4.2	Object Class Specific Services .....	15
5-4.4.2.1	Allocate Master/Slave Connection Set, Release Master/Slave Connection Set .....	16
5-4.4.2.2	Clear Diagnostics .....	16

## **5-1 Introduction**

In this standard, object modeling is used to represent the network visible behavior of devices. Devices are modeled as a collection of objects. Each class of objects is a collection of related services, attributes and behaviors. Services are the procedures that an object performs. Attributes are characteristics of objects represented by values, which can vary. An object's behavior is an indication of how the object responds to particular events.

This chapter of the specification contains the object descriptions specific to DeviceNet. The rest of the object descriptions can be found in the Volume 1, Chapter 5. With respect to the OSI reference model, CIP objects perform the Layer 7 Application functions. They also provide a mechanism to access station management counters via the network.

## **5-2 Reserved Class Codes**

The rest of the class codes are defined in the Volume 1, Chapter 5.

## **5-3 Identity Object**

### **Class Code: 03<sub>Hex</sub>**

The following table indicates modifications to the common services of the Identity Object.

**Table 5-3.1 Modifications to Identity Object Common Services**

Service Code	Need In Implementation		Service Name	Description of Service
	Class	Instance		
01 <sub>hex</sub>	Optional	Optional	Get_Attributes_All	See Volume 1, Chapter 5

## 5-4 DeviceNet Object Class Definition

### Class Code: 03<sub>Hex</sub>

The DeviceNet Object is used to provide the configuration and status of a physical attachment to DeviceNet. A product must support one (and only one) DeviceNet Object per physical network attachment as illustrated Figure 5-4.1.

Figure 5-4.1 Logical Mapping of the DeviceNet Object

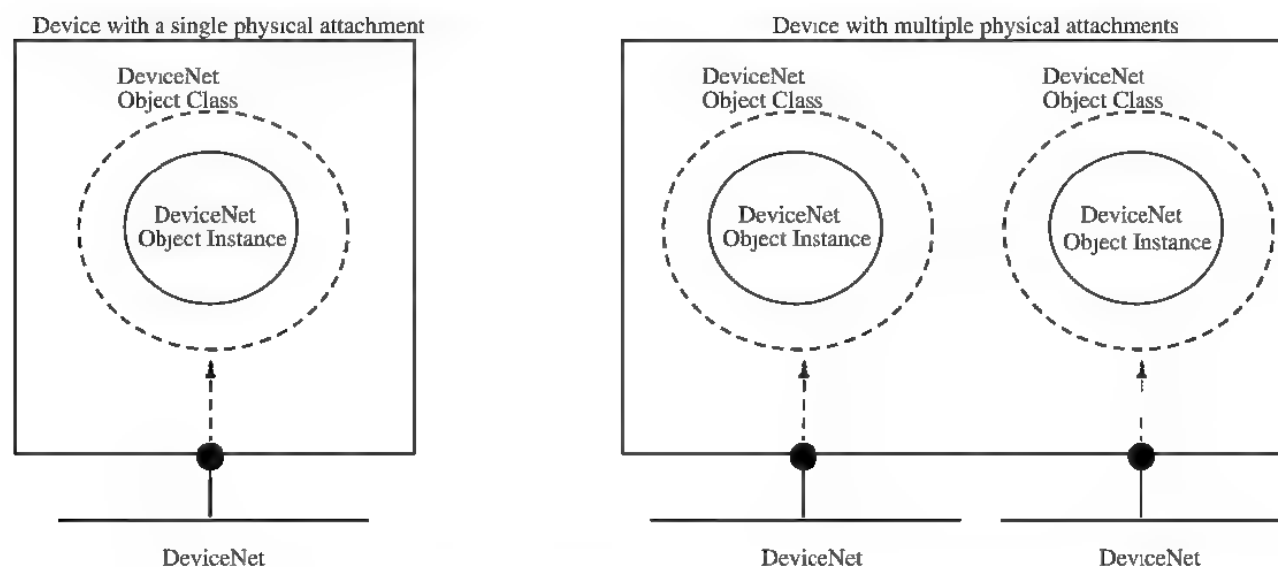


Figure 5-4.1 illustrates the fact that a logically separate DeviceNet Object Class/Instance exists per physical attachment to the network.

**Important:** Unless otherwise noted, the sections that follow assume a single physical attachment when discussing DeviceNet Object characteristics.

### 5-4.1 DeviceNet Object Class Attributes

The Class attributes for the DeviceNet Object are defined below in Table 5-4.1.

Table 5-4.1 Class Attributes for the DeviceNet Object

Attr ID	Need In Implem	Access Rule	NV	Attribute Name	Data Type	Attribute Description	Semantics of Values
1	Required	Get	NV	Revision	UINT	Revision of the DeviceNet Object Class Definition upon which the implementation is based. See description below for more details.	Range 1–65535



### 5-4.1.1 Revision - UINT data type

Defines the revision of the DeviceNet Object Class Definition upon which the implementation is based. As time progresses it is possible for technical updates of the DeviceNet Object Class Definition to occur for which an indication is desirable. This attribute enables these updates to be tracked inside devices.

The current value assigned to this attribute is two (2). If updates, which require an increase in this value are made, then the value will be increased. Support of this attribute is required.

**Table 5-4.2 Revision History**

Revision	Reason for change and description of change from previous revision.
000	Initial Definition, before specification release.
001	Initial Definition at First Release of Specification.
002	Modification of Baud Rate Attribute behavior.

### 5-4.2 DeviceNet Object Class Services

The DeviceNet Object Class supports the following CIP Common Services:

**Table 5-4.3 DeviceNet Object Class Services**

Service Code	Need In Implementation	Service Name	Service Description
0E <sub>hex</sub>	Optional	Get_Attribute_Single	Used to read a DeviceNet Object Class attribute.

### 5-4.3 DeviceNet Object Instance Attributes

Table 5-4.4 defines the Instance attributes for the DeviceNet Object.

**Table 5-4.4 Instance Attributes for the DeviceNet Object**

Attr ID	Need In Implem	Access Rule	NV	Name	Data Type	Brief Description of Attribute	Semantics of Values
1	Optional	Get/Set	NV	MAC ID	USINT	Node Address	Range 0–63
2	Optional	Get/Set	NV	Baud Rate	USINT	Baud Rate	Range 0-2
3	Optional	Get/Set	NV	BOI	BOOL	Bus-Off Interrupt	
4	Optional	Get/Set	V	Bus-Off Counter	USINT	Number of times CAN went to the bus-off state	Range 0–255
5	Optional See Note 1	Get	V	Allocation Information	STRUCT of:		
				Allocation Choice Byte	BYTE	See Section 5-4.4.1.1	
				Master's MAC ID	USINT	MAC ID of Master (from Allocate)	Range 0–63, 255 Modified via Allocate only.

Attr ID	Need In Implem	Access Rule	NV	Name	Data Type	Brief Description of Attribute	Semantics of Values
6	Conditional <sup>1</sup>	Get	V	MAC ID Switch Changed	BOOL	The Node Address Switch(es) have changed since last power-up/reset.	0 = No Change 1 = Change since last Reset or Power-up.
7	Conditional <sup>2</sup>	Get	V	Baud Rate Switch Changed	BOOL	The Baud Rate Switch(es) have changed since last power up/reset.	0 = No Change 1 = Change since last Reset or Power-up.
8	Conditional <sup>1</sup>	Get	V	MAC ID Switch Value	USINT	Actual value of Node Address switch(es)	Range 0–99
9	Conditional <sup>2</sup>	Get	V	Baud Rate Switch Value	USINT	Actual value of Baud Rate switch(es)	Range 0-9
10	Optional	Set <sup>3</sup>	NV	Quick_Connect	BOOL	Enable/Disable of Quick Connect feature	0 = Disable (default) 1 = Enable
11	Conditional <sup>4</sup>			Safety Network Number	See CIP Safety Specification (Volume 5, Chapter 5)		

Attr ID	Need In Implem	Access Rule	NV	Name	Data Type	Brief Description of Attribute	Semantics of Values
12	Optional	Get	V	Diagnostic Counters	Struct of:	List of CAN diagnostic counters	See Chapter 5-4.3.6
				Diagnostic Counters Descriptor	WORD	Indicates which Diagnostic Counters are supported	
				Arbitration Loss Count	UINT	Per CAN specification <sup>5</sup>	Range = 0 – 65,535 Default = 0
				Overload Count	UINT	Per CAN specification <sup>5</sup>	
				Bit Error Count	UINT	Per CAN specification <sup>5</sup>	
				Stuff Error Count	UINT	Per CAN specification <sup>5</sup>	
				Ack Error Count	UINT	Per CAN specification <sup>5</sup>	
				Form Error Count	UINT	Per CAN specification <sup>5</sup>	
				CRC Error Count	UINT	Per CAN specification <sup>5</sup>	
				Rx Message Loss Count	UINT	CAN subsystem has detected a lost receive message	
				Warning Error Count	UINT	Per CAN specification <sup>5</sup>	
				Rx Error Counter	UINT	Receive error counter as reported by the CAN peripheral <sup>5</sup>	Range = 0 – 256
				Tx Error Counter	UINT	Transmit error counter as reported by the CAN peripheral <sup>5</sup>	Range = 0 – 256
					UINT[5]	Reserved	Default = 0

Attr ID	Need In Implem	Access Rule	NV	Name	Data Type	Brief Description of Attribute	Semantics of Values
13	Conditional <sup>6</sup>	Get	V	Active Node Table	ARRAY of BOOL[64]	Identifies which nodes are online on the local network, based on Node Address	One bit for each node number. The node number is the index into the bit array. 0 Inactive 1 - Active

1. These attributes are required if the MAC ID switch can be set to a different value than the device is currently on-line at.
2. These attributes are required if the Baud Rate switch can be set to a different value than the device is currently on-line at.
3. This attribute shall be stored in non-volatile memory.
4. This attribute is not allowed if the device is not a safety device. If the device is a safety device, see Volume 5, CIP Safety.
5. BOSCH CAN Specification – Version 2.0, Part A. 1991, Robert Bosch GmbH.
6. This attribute is required if the device supports routing between two or more CIP ports.

#### 5-4.3.1 MAC ID

This attribute contains the MAC ID of this device. The range of values is 0 to 63 decimal.

A device that uses a switch to set the MAC ID shall return an Error Response, with General Error Code 0E<sub>hex</sub> (Attribute not settable), in response to a Set\_Attribute\_Single request when the switch setting is 0 to 63 and the MAC ID switch is enabled by other settings. When the MAC ID switch setting is greater than 63 (disabled) or the MAC ID switch is disabled by other settings, the device may support the setting of the MAC ID attribute by the Set\_Attribute\_Single service request. The device shall provide visual indication (via physical switch, LED, etc.) that the MAC ID switch has been overridden. A minor recoverable fault shall be declared when the MAC ID switch is enabled and indicates a valid MAC ID, but does not match the current on-line address of the device. During power up or reset, a device should go to the Communications Faulted state if it has a MAC ID switch set to an invalid setting and does not support the setting of the MAC ID attribute.

The MAC ID attribute is considered non-volatile in that once configured the attribute must be remembered after a power cycle or device reset. The “Out-of-box” configuration shall default to a MAC ID value of 63. The following MAC ID determination scenarios apply only to devices that use non-volatile memory to store the MAC ID of the device:

1. When the MAC ID switch is valid and enabled, the switch value is loaded into non-volatile memory when the device powers up or is reset, and prior to attempt to go on-line. The MAC ID switch value is not loaded into non-volatile memory at any other time.
2. Devices shall attempt to go on-line with the MAC ID value stored in non-volatile memory or, if in the “Out-of-box” configuration, with a MAC ID value of 63.
3. When a Set\_Attribute\_Single request with a valid MAC ID is received by a device with the MAC ID switch disabled, the non-volatile MAC ID shall be loaded with the new MAC ID.
4. When a Set\_Attribute\_Single request with a valid MAC ID is received by a device with the MAC ID switch enabled, the non-volatile MAC ID shall not change and an Error Response with General Error Code 0E<sub>hex</sub> (Attribute not settable) shall be returned.

The modification of the MAC ID requires a device to delete all Connection Objects and re-execute the Network Access State Machine defined in Chapter 2, section 2-3.

### 5-4.3.2 Baud Rate

The Baud Rate attribute indicates the selected baud rate. Values are presented in Table 5-4.5.

**Table 5-4.5 Baud Rate Attribute Values**

Value	Meaning
00	125 kbps
01	250 kbps
02	500 kbps

A device that uses a switch to set the Baud Rate shall return an Error Response, with General Error Code 0E<sub>hex</sub> (Attribute not settable), in response to a Set\_Attribute\_Single request when the Baud Rate switch is set to a valid value and not disabled by other settings. When the Baud Rate switch is not set to a valid value (disabled) and/or the Baud Rate switch is disabled by other settings, the device may support the setting of the Baud Rate attribute by the Set\_Attribute\_Single service request. The device shall provide visual indication (via physical switch, LED, etc.) that the Baud Rate switch has been overridden. A minor recoverable fault shall be declared when the Baud Rate switch is enabled and indicates a valid baud rate, but does not match the current on-line baud rate of the device. During power up or reset, a device should go to the Communication Fault state if it has a Baud Rate switch set to an invalid setting and does not support the setting of the Baud Rate attribute.

The Baud Rate attribute is considered non-volatile in that once configured the attribute must be remembered after a power cycle or device reset. The “Out-of-box” configuration shall default such that the device is able to go online at 125 kbps. The following baud rate determination scenarios apply only to devices that use non-volatile memory to store the baud rate of the device:

1. When the Baud Rate switch is valid and enabled, the switch value is loaded into non-volatile memory when the device powers up or is reset, and prior to attempting to go online. The Baud Rate switch value is not loaded into non-volatile memory at any other time.
2. Devices shall attempt to go on-line with the baud rate value stored in non-volatile memory or, if in the “Out-of-box” configuration, able to go online at 125 kbps.
3. When a Set\_Attribute\_Single request with a valid baud rate is received by a device with the Baud Rate switch disabled, the non-volatile Baud Rate shall be loaded with the new baud rate.
4. When a Set\_Attribute\_Single request with a valid baud rate is received by a device with the Baud Rate switch enabled, the non-volatile Baud Rate shall not change, and an Error Response with General Error Code 0E<sub>hex</sub> (Attribute not settable) shall be returned.

The modification of the Baud Rate will not take effect until the device is either physically reset (such as cycle of power or a reset switch) or reset by sending the Reset Service to the Identity Object. During this time the Baud Rate attribute value will not match the actual network baud rate.

### 5-4.3.3 BOI (Bus-Off Interrupt)

The BOI attribute consists of one bit that defines how a CAN device processes the bus-off interrupt. The BOI attribute is bit position 0 within a byte for the Get\_Attribute\_Single/Set\_Attribute\_Single services. The rest of the bits in the byte must be zeros.

**Table 5-4.6 BOI Attribute Values**

Value	Meaning
00	Hold the CAN chip in its bus-off (reset) state upon detection of a bus-off indication.
01	If possible, fully reset the CAN chip and continue communicating upon detection of a bus-off indication

When the BOI attribute is FALSE (set to zero) and a CAN chip bus-off event is detected, the following steps are taken:

- the CAN chip is held in its reset/bus-off state
- the device enters the Communications Faulted state (see Chapter 2, section 2-3.1).

When the BOI attribute is TRUE (set to one) and a CAN chip bus-off event is detected, it *may* be possible to return the CAN chip to its normal operating mode and continue communicating based on the Network Access State Machine presented in Chapter 2, section 2-3.

**Important:** If the BOI attribute is set to one the device **must** insure that it does not perpetually reset and continue to produce corrupted packets on the bus. Failing to do so will allow the device to disrupt all communications on the bus.

Connections are not necessarily effected when a bus-off event is detected and the device is able to continue communicating. Previously established connections can remain existing or they can be deleted (soft reset). Either way, the Duplicate MAC ID detection algorithm must be performed again per the Network Access State Machine.

As indicated in Table 5-4.4, support of the BOI attribute is optional. If it is not supported, a device must implement the behavior indicated by attribute value zero (0) in Table 5-4.6.

#### **5-4.3.4 Bus-off Counter**

The Bus-off Counter counts the number of times the CAN chip went to the bus-off state (counts number of bus-off interrupts). The counter has values of 0 to 255 decimal.

The Bus-off Counter is initialized to zero at power-up or device initialization.

The Bus-off Counter stops counting when it reaches maximum count. The counter does NOT roll over. The Counter stays at maximum count until a Set\_Attribute\_Single is performed.

The DeviceNet Object resets the Bus-off Counter to zero (0) whenever it receives a Set\_Attribute\_Single request specifying the Bus-Off Counter attribute. The Set\_Attribute\_Single data is not used and can be any value. The transmission of a Set\_Attribute\_Single request to the Bus-off Counter is all that's required to reset the counter.

#### **5-4.3.5 Allocation Information**

The Allocation Information attribute is pertinent to the Predefined Master/Slave Connection Set. Its support is required if the Predefined Master/Slave Connection Set is supported. It indicates whether or not the Predefined Master/Slave Connection Set defined in Chapter 3, section 3-6 has been allocated. If it has been allocated, then, this attribute indicates the device that has performed the allocation and the Connection(s) that are currently allocated.

This attribute is modified when a successful response associated with an Allocate\_Master/Slave\_Connection\_Set service (defined later in this section) is generated. This attribute is not modifiable by the Set\_Attribute\_Single Service. An Error Response whose General Error Code Field is set to 0E<sub>hex</sub> (Attribute not settable) is returned if a Set\_Attribute\_Single request specifies this attribute.

The Allocation Information attribute consists of the following:

##### **5-4.3.5.1 Allocation Choice Byte**

The Allocation Choice byte indicates which of the Predefined Master/Slave Connections are *active* (in the Configuring, or Established state). Its format is specified in Chapter 3, section 3-5.1.

The Allocation Choice byte is initialized to 00 at device power up or reset.

**5-4.3.5.2 Master's MAC ID**

The Master's MAC ID contains the MAC ID of the device that has allocated the Predefined Master/Slave Connection Set via the Allocate\_Master/Slave\_Connection\_Set service. This contains the *Allocator's MAC ID* field copied from the Allocate\_Master/Slave\_Connection\_Set request.

The range of values is 0 to 63 and 255 decimal. A value in the range 0–63 indicates that the Predefined Master/Slave Connection Set is currently allocated and denotes the MAC ID of the device that performed the allocation. The value 255 means the Predefined Master/Slave Connection set has not been allocated.

The Master's MAC ID attribute is initialized to 255 (FF hex) at device power up/reset.

**5-4.3.6 Diagnostic Counters**

This attribute reports the counts of various types of network errors. The first field, Diagnostic Counters Descriptor, identifies which counters are supported by the device. Each bit indicates if a designated counter is supported. If set, the counter is supported.

**Table 5-4.7 Diagnostic Counters Bit Description**

Bit	Counter	Clearable
0	Arbitration Loss	Yes
1	Overload Error	Yes
2	Bit Error	Yes
3	Stuff Error	Yes
4	Ack Error	Yes
5	Form Error	Yes
6	CRC Error	Yes
7	Rx Message Loss	Yes
8	Warning Error	Yes
9	Rx Error	No
10	Tx Error	No
11 – 15	Reserved (shall be 0)	N/A

The remaining fields provide counter values for each supported counter. The counters are advanced by one for each occurrence of the error, except for the Rx and Tx Error Counters (fields 10 and 11) which are incremented and decremented by the CAN peripheral based on the CAN specification, and do not roll over.

All counters are cleared to zero when the device enters the Sending Duplicate MAC ID Check Request Message (see Network Access State Transition Diagram in Chapter 2).



**5-4.3.7 Active Node Table**

The Active Node Table attribute reports the activity state of each node on the network, based on node number. Each bit in the array represents a node on the local network, with the bit position matching the node address represented (i.e. Bit 0 = Node Address 0, Bit 1 = Node Address 1, etc.). When the bit is set (TRUE) the node is active on the link. When the bit is cleared (FALSE) the node is inactive on the link.

The bit is set (node is indicated as active) when the node represented by the bit is in either the On-Line, Sending Duplicate MAC ID Check Request Message, or Waiting for Duplicate MAC ID Check Message state. The setting of the bit shall occur within 1 second of a previously inactive node transmitting on the link. Also, the device containing the Active Node Table shall set all bits when transitioning to the On-Line state.

The bit is cleared (node is indicated as inactive) when the node represented by the bit is in either the Non-Existent or Communication Fault state. The clearing of the bit shall occur within 20 seconds of the previously active node leaving the link (or, if the node is not present on the network, within 20 seconds after the device containing the Active Node Table transitions to the On-Line state). Also, the device containing the Active Node Table shall clear all bits when transitioning out of the On-Line state.

A router shall not attempt to route an unconnected message request to a node on the local network when the Active Node Table bit for that node indicates inactive (bit is cleared). Instead, the router shall immediately return a General Error status of 0x01 and an Extended Error status of 0x0204.

**5-4.4 DeviceNet Object Instance Services**

The sections that follow describe the Common Services and Object Class Specific Services supported by the DeviceNet Object Instance.

**5-4.4.1 Common Services**

The DeviceNet Object Instance supports the following Common Services:

**Table 5-4.8 DeviceNet Object Instance Common Services**

Service Code	Need In Implementation	Service Name	Service Description
05 <sub>hex</sub>	Conditional	Reset	Used to reset this instance of the DeviceNet object
0E <sub>hex</sub>	Optional	Get_Attribute_Single	Used to read a DeviceNet Object attribute value.
10 <sub>hex</sub>	Optional	Set_Attribute_Single	Used to modify a DeviceNet Object attribute value.

1 -- This service is required when this DeviceNet object instance is addressable through some other CIP access mechanism other than the subnet interface controlled by this DeviceNet object instance.

#### 5-4.4.1.1 Reset Service

When the DeviceNet Object receives a Reset request, it:

- determines if it can provide the type of reset requested
- responds to the request
- perform the type of reset requested

The Reset common service has the following object-specific parameter:

**Table 5-4.9 Reset Service Parameter**

Name	Type	Description of Request Parameters	Semantics of Values
Type	USINT	Type of Reset	See Table below.

The parameter Type for the Reset common service has the following enumeration specifications:

**Table 5-4.10 Reset Service Parameter Values**

Value:	Type of Reset:
0	Emulate as closely as possible cycling power on the network link that the <i>DeviceNet Object</i> instance represents. This value is the default if this parameter is omitted.
1	Return as closely as possible to the out-of-box configuration, then emulate cycling power on the network link as closely as possible.
2	Excepting the MAC ID and Baud Rate, return as closely as possible to the out-of-box configuration, then emulate cycling power on the network link as closely as possible. The MAC ID and Baud Rate is not changed by this reset.
3 thru 99	Reserved
100 thru 199	Vendor specific reset behavior
200 thru 255	Reserved

#### 5-4.4.2 Object Class Specific Services

The DeviceNet Object Instance also supports the following Object Class Specific Services:

**Table 5-4.11 Class Specific Services Provided by the DeviceNet Object**

Service Code	Need In Implementation	Service Name	Service Description
4B <sub>hex</sub>	Optional	Allocate_Master/Slave_Connection_Set	Requests the use of the Predefined Master/Slave Connection Set.
4C <sub>hex</sub>	Conditional <sup>1</sup>	Release_Master/Slave_Connection_Set	Indicates that the specified Connections within the Predefined Master/Slave Connection Set are no longer desired. These Connections are to be released (Deleted)
4D <sub>hex</sub>	Conditional <sup>2</sup>	Clear Diagnostics	Clears the diagnostic counters that are reported in Attribute 12, Diagnostic Counters

1—The Release\_Master/Slave\_Connection\_Set service is required if the Allocate\_Master/Slave\_Connection\_Set service is implemented in the device.

2—This service shall be supported if any clearable counters are implemented in the Diagnostic Counters attribute.

#### 5-4.4.2.1 Allocate Master/Slave Connection Set, Release Master/Slave Connection Set

These services are used to allocate and deallocate the Predefined Master/Slave Connection Set described in Chapter 3, section 3-6.

A device that behaves as the Client across the Predefined Master/Slave Connection Set is referred to as a *Master*. A device that behaves as the Server across the Predefined Master/Slave Connection Set is referred to as a *Slave*. Within the bounds of the service descriptions to follow, a *Master* and/or *Slave* is viewed as a functional unit within a communicating device.

A device that wants to function as another's Master must first *allocate* the Predefined Master/Slave Connection Set within the Slave. Only **one** Master can have the Predefined Master/Slave Connection Set allocated at any given time. The entire Connection Set is allocated and the Master uses a select subset of the Connections from the set (i.e. Bit Strobe only, or Poll only). When a Master wants to "give-up" its Slave it *releases* all connections, causing the Slave to "deallocate" the Predefined Master/Slave Connection Set.

#### 5-4.4.2.2 Clear Diagnostics

This service clears (to zero) the clearable counters within Attribute 12 (Diagnostic Counters). The counters that are clearable are indicated in Table 5-4.7/ Table 5-4.7}.

## **Volume 3: DeviceNet Adaptation of CIP**

# **Chapter 6: Device Profiles**

---

## **Contents**

6-1	Introduction.....	3
6-2	Required Objects.....	3

## 6-1 Introduction

This chapter of the DeviceNet application contains additions to the CIP object model that are DeviceNet specific. At this time, no such additions exist.

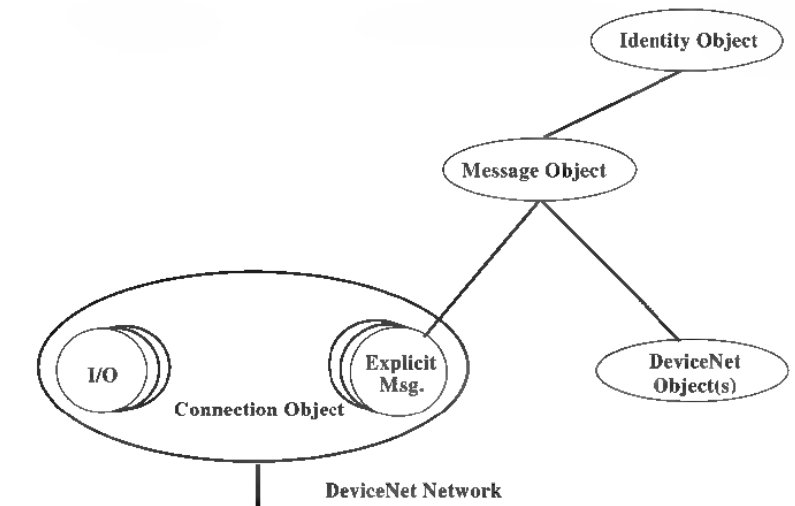
## 6-2 Required Objects

At minimum, every DeviceNet device shall implement instance number one of the following objects:

**Table 6-2.1 Required Objects**

Object Class	Optional/Required	# of Instances
Identity	Required	1
Message Router	Required	1
DeviceNet Object	Required	1
Connection	Required	2 (explicit,-I/O)

**Figure 6-2.1 Base Device Object Model**



This page is intentionally left blank

## **Volume 3: DeviceNet Adaptation of CIP**

# **Chapter 7: Electronic Data Sheets**

---



## Contents

7-1	Introduction.....	4
7-2	EDS Content .....	4
7-3	Device Description Section [Device] .....	4
7-4	[Device Classification] Section .....	5
7-5	I/O Characteristics Section.....	6
7-5.1	Default I/O Information Entry .....	7
7-5.1.1	Default I/O Type Mask .....	7
7-5.2	Poll Information Entry .....	7
7-5.2.1	Compatible I/O Type Mask.....	8
7-5.2.2	Default Producing Connection .....	8
7-5.2.3	Default Consuming Connection .....	8
7-5.3	Strobe Information Entry .....	8
7-5.3.1	Compatible I/O Type Mask.....	8
7-5.3.2	Default Producing Connection .....	9
7-5.3.3	Default Consuming Connection .....	9
7-5.4	Change of State Information Entry .....	9
7-5.4.1	Compatible I/O Type Mask.....	9
7-5.4.2	Default Producing Connection .....	9
7-5.4.3	Default Consuming Connection .....	10
7-5.5	Cyclic Information Entry .....	10
7-5.5.1	Compatible I/O Type Mask.....	10
7-5.5.2	Default Producing Connection .....	10
7-5.5.3	Default Consuming Connection .....	11
7-5.6	Multicast Poll Information Entry ....	11
7-5.6.1	Compatible I/O Type Mask.....	11
7-5.6.2	Default Producing Connection .....	11
7-5.6.3	Default Consuming Connection .....	11
7-5.7	Device's Producing Connection Entries .....	12
7-5.7.1	Size.....	13
7-5.7.2	Number of Significant Bits .....	13
7-5.7.3	Compatible I/O Type Mask.....	13
7-5.7.4	Name String .....	13
7-5.7.5	Connection Path Size .....	13
7-5.7.6	Connection Path .....	14
7-5.7.7	Help String .....	14
7-5.8	Device's Consuming Connection Entries .....	14
7-5.8.1	Size.....	15
7-5.8.2	Number of Significant Bits .....	15
7-5.8.3	Compatible I/O Type Mask.....	15
7-5.8.4	Name String .....	15
7-5.8.5	Connection Path Size .....	15
7-5.8.6	Connection Path .....	16
7-5.8.7	Help String .....	16
7-5.9	Examples.....	16
7-5.9.1	Poll & Strobe Capable Device, Poll is Default.....	16
7-5.9.2	Strobe Only Device, Only One Input and One Output..	18
7-5.9.3	Poll & Change of State Capable Device, Poll and Change of State is Default .....	19
7-5.9.4	Example Showing Use of Extended IO Info Keywords.....	20
7-6	Variant I/O Info Section.....	24
7-7	Parameter Enumeration Strings Section.....	27
7-8	Internationalization Section .....	28
7-8.1	Input Entry Keywords.....	28
7-8.1.1	International Name, Field 1.....	28

7-8.1.2	International Help String, Field 2.....	28
7-8.2	Output Entry Keywords .....	28
7-8.2.1	International Name, Field 1.....	28
7-8.2.2	International Help String, Field 2.....	28
7-8.3	Selection and EndSelection Entry Keywords.....	29

## 7-1 Introduction

This chapter of the DeviceNet specification contains additions to the definition of electronic data sheets (EDS) that are DeviceNet specific. See the CIP Common specification for more information about the format of electronic data sheets and the definition of EDS related terms such as EDS section, EDS entry and EDS field.

## 7-2 EDS Content

The structure of the sections, legal section delimiters and ordering of sections in an EDS is a combination of what is specified in Volume 1, Chapter 7-3.3.2 and the DeviceNet-specific sections listed here.

**Table 7-2.1 DeviceNet-specific Sections of an EDS File**

Section Name	Legal Delimiter	Placement	Required/Optional
I/O Characteristics	[IO_Info]	*	Conditional
Enumerated Parameters	[EnumPar]	*	Optional
Variant I/O Characteristics	[Variant_IO_Info]	*	Conditional

## 7-3 Device Description Section [Device]

In addition to the sections shown in Table 7-3.12 in Volume 1, Chapter 7-3.6.2, the following DeviceNet-specific entries are defined.

**Table 7-3.1 Fields in the Device Description Section**

Entry Name	Entry Keyword <sup>1</sup>	Field Number	Data Type	Required/Optional
DeviceNet Quick Connect	DNetQC	1	WORD	Optional
		2	UINT	Optional

- **DeviceNet Quick Connect** – The bit values in the first field identify the type(s) of Quick Connect supported by this device as shown in Table 7-3.2 (a value of zero, or the absence of this entry keyword, indicates Quick Connect is not supported). The second field indicates the time, in milliseconds, from the application of device power to the production of the first Duplicate MACID Request message for this device.

**Table 7-3.2 DeviceNet Quick Connect Support Type Bit Field**

Bit	Bit Definition
0	Quick Connect supported at power up
1	Quick Connect supported as a Client (connection originating) device
2 – 15	Reserved

An example usage of the DeviceNet Quick Connect entry in the Device Description section of an EDS file is shown in Figure 7-4.1.

## 7-4 [Device Classification] Section

See Volume 1, Chapter 7-3.5.3 Device Classification Section for description of this section of the EDS file.

This section is conditional for DeviceNet EDS files. DeviceNet EDS files that contain a [Modular] section are required to include the [Device Classification] section. For all other DeviceNet EDS files the [Device Classification] section is optional. If the [Device Classification] section is not found in an EDS file and there is no [Modular] section, the device shall be considered to be a DeviceNet product. When the [Device Classification] section exists within the EDS, for any DeviceNet compliant device, there shall be at least one ClassN keyword entry with its first field set to DeviceNet. As shown in Figure 7-4.1, no sub-classifications shall be present.

**Figure 7-4.1 Example of partial EDS of a DeviceNet device**

```
[File]
  DescText = "Widget EDS File";
  CreateData = 02 07 2001;
  CreateTime = 17:51:44;
  ModDate   04-06-1997;
  ModTime   22:07:30;
  Revision = 2.1;
  HomeURL   "http://www.myedsfiles.com/EDS/12345.eds";

[Device]
  VendCode = 65535;
  VendName  "Widget-Works, Inc.";
  ProdType = 43;
  ProdTypeStr = "Generic";
  ProdCode = 10;
  MajRev = 1;
  MinRev = 1;
  ProdName  "Smart-Widget";
  Catalog   "6666-LL";
  Icon      "widget.ico";
  DNetQC = 0b0000000000000001, 500; $ Quick Connect supported at powerup,
                                     $ 500ms power up time.

[Device Classification]
  Class1 = DeviceNet;
```

## 7-5 I/O Characteristics Section

The [IO\_Info] section contains information about a device's Predefined Master/Slave Connection Set (see Chapter 3) I/O capabilities. This section is required if a device implements any IO Connection whose characteristics can be completely represented with one or more entries within this section. This section describes the following I/O characteristics of a device:

1. I/O Trigger Type: Poll, Multicast Poll, Strobe, Change of State and Cyclic
2. I/O Message Size: Number of bytes
3. Predefined I/O Connection Paths

Table 7-5.1 shows the entries in the [IO\_Info] section.

**Table 7-5.1 IO\_Info Section Entries**

Entry Name	Entry Keyword	Required/Optional
Default I/O Information	Default	Required
Poll Information	PollInfo or PollInfoExa	Conditional
Strobe Information	StrobeInfo or StrobeInfoExa	Conditional
Multicast Poll Information	MulticastPollInfo or MulticastPollInfoExa	Conditional
Change of State Information	COSInfo or COSInfoExa	Conditional
Cyclic Information	CyclicInfo or CyclicInfoExa	Conditional
Device's Producing Connection	Input or InputExa	Conditional
Device's Consuming Connection	Output or OutputExa	Conditional

The entries in the [IO\_Info] section provide:

- Default I/O Information - Defines the default I/O information of the device.
- Poll Information - Defines the I/O types that can be used in conjunction with the Poll connection and the default Poll Producing and Consuming connections. If this entry is not present, the polled connection is not supported in the device.
- Strobe Information - Defines the I/O types that can be used in conjunction with the Strobe connection and the default Strobe Producing and Consuming connections. If this entry is not present, the strobe connection is not supported in the device.
- Change of State Information - Defines the I/O types that can be used in conjunction with the Change of State connection and the default Change of State Producing and Consuming connections. If this entry is not present, the Change of State connection is not supported in the device.
- Cyclic Information - Defines the I/O types that can be used in conjunction with the Cyclic connection and the default Cyclic Producing and Consuming connections. If this entry is not present, the Cyclic connection is not supported in the device.
- Multicast Poll Information - Defines the I/O types that can be used in conjunction with the Multicast Poll connection and the default Multicast Poll Producing and Consuming connections. If this entry is not present, the multicast poll connection is not supported in the device.
- Device's Producing Connection - Defines the byte and bit size of the connection, which I/O types it can be used with, the connection path, a name string and a help string.
- Device's Consuming Connection - Defines the byte and bit size of the connection, which I/O types it can be used with, the connection path, a name string and a help string.

### 7-5.1 Default I/O Information Entry

The Default I/O Information Entry defines the default I/O information of the device. This entry is delimited by **Default=**.

The fields of the Default Information Entry are defined in Table 7-5.2.

**Table 7-5.2 Fields for the “Default=” Entry**

Field Name	Field Number	Data Type	Required/Optional
Default I/O Type Mask	1	WORD	Required

#### 7-5.1.1 Default I/O Type Mask

A bit mapped field that defines the I/O types to be enabled as a default for the device. If a bit is set to one, that type of connection is to be enabled as a default. The bit assignments are defined in Table 7-5.3.

**Table 7-5.3 Default I/O Type Mask Bit Definitions**

Bit	Bit Definition
0	Poll
1	Strobe
2	Change of State
3	Cyclic
4	Multicast Poll
5-15	Reserved

### 7-5.2 Poll Information Entry

The Poll Information Entry defines the I/O types that can be used in conjunction with the Poll connection. It also defines the default Poll Input and Output connections. This entry is delimited by **PollInfo=** or **PollInfoExa=**. If a configuration tool supports the “PollInfoExa” keyword and a “PollInfoExa” keyword exists in the EDS the “PollInfo” keyword shall be ignored. If this entry is not present, then:

- The Poll connection is not supported in the device or
- The connection cannot be completely described by the constructs of this keyword.

The fields of the Poll Information entry are defined in Table 7-5.4.

**Table 7-5.4 Fields for the “PollInfo=” and “PollInfoExa” Entries**

Field Name	Field Number	Data Type	Required/Optional
Compatible I/O Type Mask	1	WORD	Required
Default Producing Connection	2	UINT	Required
Default Consuming Connection	3	UINT	Required

### 7-5.2.1 Compatible I/O Type Mask

A bit mapped field that defines the I/O types that the polled connection can be combined with. The bit assignments are the same as the Default Type Mask. The poll bit should be set.

### 7-5.2.2 Default Producing Connection

Specifies one of the Producing Connection Entries as the default for the polled connection. A zero indicates that there is no producing data for the poll connection.

If “PollInfo” is the specified keyword the Poll Information entry refers to an InputN keyword. If “PollInfoExa” is the specified keyword the Poll Information entry refers to an InputExaN keyword.

### 7-5.2.3 Default Consuming Connection

Specifies one of the Consuming Connection Entries as the default for the polled connection. A zero indicates that there is no consuming data for the poll connection.

If “PollInfo” is the specified keyword the Poll Information entry refers to an OutputN keyword. If “PollInfoExa” is the specified keyword the Poll Information entry refers to an OutputExaN keyword.

## 7-5.3 Strobe Information Entry

The Strobe Information Entry defines the I/O types that can be used in conjunction with the Strobe connection. It also defines the default Strobe Producing and Consuming connections. This entry is delimited by **StrobeInfo=** or **StrobeInfoExa=**. If a configuration tool supports the “StrobeInfoExa” keyword and a “StrobeInfoExa” keyword exists in the EDS the “StrobeInfo” keyword shall be ignored. If this entry is not present, then

- The Strobe connection is not supported in the device or
- The connection can not be completely described by the constructs of this keyword

The fields of the Strobe Information Entry are defined in Table 7-5.5.

**Table 7-5.5 Fields for the “StrobeInfo=” and “StrobeInfoExa” Entries**

Field Name	Field Number	Data Type	Required/Optional
Compatible I/O Type Mask	1	WORD	Required
Default Producing Connection	2	UINT	Required
Default Consuming Connection	3	UINT	Required

### 7-5.3.1 Compatible I/O Type Mask

A bit mapped field that defines the I/O types that the strobed connection can be combined with. The bit assignments are the same as the Default Type Mask. The strobe bit should be set.

### 7-5.3.2 Default Producing Connection

Specifies one of the Producing Connection Entries as the default for the strobed connection. A zero indicates that there is no producing data for the strobe connection.

If “StrobeInfo” is the specified keyword the Strobe Information entry refers to an InputN keyword. If “StrobeInfoExa” is the specified keyword the Strobe Information entry refers to an InputExaN keyword.

### 7-5.3.3 Default Consuming Connection

Specifies one of the consuming Connection Entries as the default for the strobed connection. A zero indicates that there is no consuming data for the strobe connection.

If “StrobeInfo” is the specified keyword the Strobe Information entry refers to an OutputN keyword. If “StrobeInfoExa” is the specified keyword the Strobe Information entry refers to an OutputExaN keyword.

## 7-5.4 Change of State Information Entry

The Change of State Information Entry defines the I/O types that can be used in conjunction with the Change of State connection. It also defines the default Change of State Input and Output connections. This entry is delimited by **COSInfo=** or **COSInfoExa=**. If a configuration tool supports the “COSInfoExa” keyword and a “COSInfoExa” keyword exists in the EDS the “COSInfo” keyword shall be ignored. If this entry is not present, then:

- The Change of State connection is not supported in the device or
- The connection cannot be completely described by the constructs of this keyword.

The fields of the Change of State Information Entry are defined in Table 7-5.6.

**Table 7-5.6 Fields for the “COSInfo=” and “COSInfoExa” Entries**

Field Name	Field Number	Data Type	Required/Optional
Compatible I/O Type Mask	1	WORD	Required
Default Producing Connection	2	UINT	Required
Default Consuming Connection	3	UINT	Required

### 7-5.4.1 Compatible I/O Type Mask

A bit mapped field that defines the I/O types that the Change of State connection can be combined with. The bit assignments are the same as the Default Type Mask. The Change of State bit should be set.

### 7-5.4.2 Default Producing Connection

Specifies one of the Producing Connection Entries as the default for the Change of State connection. A zero indicates that there is no producing data for the Change of State connection.



If “COSInfo” is the specified keyword the Change of State Information entry refers to an InputN keyword. If “COSInfoExa” is the specified keyword the Change of State Information entry refers to an InputExaN keyword.

#### 7-5.4.3 Default Consuming Connection

Specifies one of the Consuming Connection Entries as the default for the Change of State connection. A zero indicates that there is no consuming data for the Change of State connection.

If “COSInfo” is the specified keyword the Change of State Information entry refers to an OutputN keyword. If “COSInfoExa” is the specified keyword the Change of State Information entry refers to an OutputExaN keyword.

#### 7-5.5 Cyclic Information Entry

The Cyclic Information Entry defines the I/O types that can be used in conjunction with the Cyclic connection. It also defines the default Cyclic Producing and Consuming connections. This entry is delimited by **CyclicInfo=** or **CyclicInfoExa=**. If a configuration tool supports the “CyclicInfoExa” keyword and a “CyclicInfoExa” keyword exists in the EDS the “CyclicInfo” keyword shall be ignored. If this entry is not present, then:

- The Cyclic connection is not supported in the device or
- The connection cannot be completely described by the constructs of this keyword.

The fields of the Cyclic Information Entry are defined in Table 7-5.7.

**Table 7-5.7 Fields for the “CyclicInfo=” and “CyclicInfoExa=” Entries**

Field Name	Field Number	Data Type	Required/Optional
Compatible I/O Type Mask	1	WORD	Required
Default Producing Connection	2	UINT	Required
Default Consuming Connection	3	UINT	Required

##### 7-5.5.1 Compatible I/O Type Mask

A bit mapped field that defines the I/O types that the Cyclic connection can be combined with. The bit assignments are the same as the Default Type Mask. The Cyclic bit should be set.

##### 7-5.5.2 Default Producing Connection

Specifies one of the Producing Connection Entries as the default for the Cyclic connection. A zero indicates that there is no producing data for the Cyclic connection.

If “CyclicInfo” is the specified keyword the Cyclic Information entry refers to an InputN keyword. If “CyclicInfoExa” is the specified keyword the Cyclic Information entry refers to an InputExaN keyword.

### 7-5.5.3 Default Consuming Connection

Specifies one of the consuming Connection Entries as the default for the Cyclic connection. A zero indicates that there is no consuming data for the Cyclic connection.

If “CyclicInfo” is the specified keyword the Cyclic Information entry refers to an OutputN keyword. If “CyclicInfoExa” is the specified keyword the Cyclic Information entry refers to an OutputExaN keyword.

### 7-5.6 Multicast Poll Information Entry

The Multicast Poll Information Entry defines the I/O types that can be used in conjunction with the Multicast Poll connection. It also defines the default Multicast Poll Producing and Consuming connections. This entry is delimited by **MulticastPollInfo=** or **MulticastPollInfoExa=**. If a configuration tool supports the “MulticastPollInfoExa” keyword and a “MulticastPollInfoExa” keyword exists in the EDS the “MulticastPollInfo” keyword shall be ignored. If this entry is not present, then:

- The Multicast Poll connection is not supported in the device or
- The connection cannot be completely described by the constructs of this keyword.

The fields of the Multicast Poll Information Entry are defined in Table 7-5.8.

**Table 7-5.8 Fields for the “MulticastPollInfo=” and “MulticastPollInfoExa” Entries**

Field Name	Field Number	Data Type	Required/Optional
Compatible I/O Type Mask	1	WORD	Required
Default Producing Connection	2	UINT	Required
Default Consuming Connection	3	UINT	Required

#### 7-5.6.1 Compatible I/O Type Mask

A bit mapped field that defines the I/O types that the Multicast Poll connection can be combined with. The bit assignments are the same as the Default Type Mask. The Multicast Poll bit should be set.

#### 7-5.6.2 Default Producing Connection

Specifies one of the Producing Connection Entries as the default for the Multicast Poll connection. A zero indicates that there is no producing data for the Multicast Poll connection.

If “MulticastPollInfo” is the specified keyword the Multicast Poll Information entry refers to an InputN keyword. If “MulticastPollInfoExa” is the specified keyword the Multicast Poll Information entry refers to an InputExaN keyword.

#### 7-5.6.3 Default Consuming Connection

Specifies one of the consuming Connection Entries as the default for the Multicast Poll connection. A zero indicates that there is no consuming data for the Multicast Poll connection.

If “MulticastPollInfo” is the specified keyword the Multicast Poll Information entry refers to an OutputN keyword. If “MulticastPollInfoExa” is the specified keyword the Multicast Poll Information entry refers to an OutputExaN keyword.

### 7-5.7 Device’s Producing Connection Entries

The Device’s Producing Connection Entries are described using the “Input” or “InputExa” keyword. The “Input” or “InputExa” keyword shall be used to define the logical encoded path to the data of one or more objects that can be produced by an IO Connection. The “Input” or “InputExa” keyword shall be required for each occurrence:

- That is capable of being set in the produced connection path and
- Can be described within an EDS file.

Each entry defines the byte and bit size of the connection, which I/O types it can be used with, the connection path, a name string and a help string. The delimiter for the Producing Connection Entries consists of the string “Input” or “InputExa” combined with a decimal number, for example “Input1=” or “InputExa1=”. For “Input” the decimal numbers must start at one and increment by one. For “InputExa”, the decimal numbers must start at one and increment by one. The entry for “InputN” contains the fields defined in Table 7-5.9. The entry for “InputExaN” contains the field defined in Table 7-5.10.

If the “InputExa” form of this keyword is used to describe the Device’s Producing connections the “Input” form of this keyword shall also be used to describe the Device’s Producing connections. If a configuration tool supports the “InputExa” keyword and one or more “InputExaN” keywords exist in the EDS then all “InputN” keywords shall be ignored.

**Table 7-5.9 Fields for the “InputN=” Entry**

Field Name	Field Number	Data Type	Required/Optional
Size	1	UINT	Required
Number of Significant Bits	2	UINT	Required
Compatible I/O Type Mask	3	WORD	Required
Name String	4	ASCII Character Array	Required
Connection Path Size	5	UINT	Required
Connection Path	6	EPATH	Required
Help String	7	ASCII Character Array	Required

**Table 7-5.10 Fields for the “InputExaN=” Entry**

Field Name	Field Number	Data Type	Required/Optional
Size	1	UINT or a ParamN reference	Conditional <sup>1</sup>
Number of significant bits	2	UINT	Conditional <sup>2</sup>
Compatible I/O Type Mask	3	WORD	Required
Name String	4	ASCII Character Array	Conditional <sup>2</sup>
Connection Path Size	5	UINT	Conditional <sup>1</sup>
Connection Path	6	EPATH, ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference	Required
Help String	7	ASCII Character Array	Conditional <sup>2</sup>

<sup>1</sup> - Optional if a ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference is used in the ‘Connection Path’ field

<sup>2</sup> - Optional if an VariantN or VariantExaN reference is used in the ‘Connection Path’ field

#### 7-5.7.1 Size

The size in bytes of this data. For the “InputExaN” form of this keyword, if this field is empty then the size shall be determined from the ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference in the ‘Connection Path’ field. For the “InputExaN” form of this keyword, this entry can either be a numerical value or a reference to a ParamN entry keyword. If a ‘ParamN’ reference is specified, then the data type of that parameter shall be a UINT.

#### 7-5.7.2 Number of Significant Bits

The number of consecutive significant bits of this data (bits that are actually used). A zero indicates that all bits of the data are significant.

#### 7-5.7.3 Compatible I/O Type Mask

A bit mapped field that defines the I/O types that this data can be used with. The bit assignments are the same as the Default Type Mask.

#### 7-5.7.4 Name String

The textual data name. (32 characters max, when displayed truncation may occur to meet the display capabilities.)

#### 7-5.7.5 Connection Path Size

The number of bytes used to represent the path. For the “InputExaN” form of this keyword, if this field is empty then the connection path size shall be determined from the ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference in the ‘Connection Path’ field.

### 7-5.7.6 Connection Path

The connection path of the data. For the “InputExaN” form of this keyword, if a ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference is specified then the ‘Connection Path’ is the path specified in the ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference.

### 7-5.7.7 Help String

The textual help string. When displayed, truncation may occur to meet the display capabilities.

## 7-5.8 Device’s Consuming Connection Entries

The Device’s Consuming Connection Entries are described using the “Output” or “OutputExa” keyword. The “Output” or “OutputExa” keyword shall be used to define the logical encoded path from the data of one or more objects that can be consumed by an IO Connection. The “Output” or “OutputExa” keyword shall be required for each occurrence:

- That is capable of being set in the consumed connection path and
- Can be described within an EDS file.

Each entry defines the byte and bit size of the connection, which I/O types it can be used with, the connection path, a name string and a help string. The delimiter for the Consumer Connection Entries consists of the string “Output” or “OutputExa” combined with a decimal number, for example “Output1=” or “OutputExa1=” For “Output”, the decimal numbers must start at one and increment by one. For “OutputExa”, the decimal numbers must start at one and increment by one. The entry for “OutputN” contains the fields defined in Table 7-5.11. The entry for “OutputExaN” contains the fields defined in Table 7-5.12.

If the “OutputExa” form of this keyword is used to describe the Device’s Consuming connections the “Output” form of this keyword shall also be used to describe the Device’s Consuming connections. If a configuration tool supports the “OutputExa” keyword and one or more “OutputExaN” keywords exist in the EDS then all “OutputN” keywords shall be ignored.

**Table 7-5.11 Fields for the “OutputN=” Entry**

Field Name	Field Number	Data Type	Required/Optional
Size	1	UINT	Required
Number of Significant Bits	2	UINT	Required
Compatible I/O Type Mask	3	WORD	Required
Name String	4	ASCII Character Array	Required
Connection Path Size	5	UINT	Required
Connection Path	6	EPATH	Required
Help String	7	ASCII Character Array	Required

**Table 7-5.12 Fields for the “OutputExaN=” Entry**

Field Name	Field Number	Data Type	Required/Optional
Size	1	UINT or a ParamN reference	Conditional <sup>1</sup>
Number of significant bits	2	UINT	Conditional <sup>2</sup>
Compatible I/O Type Mask	3	WORD	Required
Name String	4	ASCII Character Array	Conditional <sup>2</sup>
Connection Path Size	5	UINT	Conditional <sup>1</sup>
Connection Path	6	EPATH, ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference	Required
Help String	7	ASCII Character Array	Conditional <sup>2</sup>

1 – Optional if a ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference is used in the ‘Connection Path’ field

2 – Optional if a VariantN or VariantExaN reference is used in the ‘Connection Path’ field.

#### 7-5.8.1 Size

The size in bytes of this data. For the “OutputExaN” form of this keyword, if this field is empty then the size shall be determined from the ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference in the ‘Connection Path’ field. For the “OutputExaN” form of this keyword, this entry can either be a numerical value or a reference to a ParamN entry keyword. If a ‘ParamN’ reference is specified, then the data type of that parameter shall be a UINT.

#### 7-5.8.2 Number of Significant Bits

The number of consecutive significant bits of this data (bits that are actually used). A zero indicates that all bits of the data are significant.

#### 7-5.8.3 Compatible I/O Type Mask

A bit mapped field that defines the I/O types that this data can be used with. The bit assignments are the same as the Default Type Mask.

#### 7-5.8.4 Name String

The textual data name. (32characters max, when displayed, truncation may occur to meet the display capabilities of the configuration tool)

#### 7-5.8.5 Connection Path Size

The number of bytes used to represent the path. For the “OutputExaN” form of this keyword, if this field is empty then the connection path size shall be determined from the ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference in the ‘Connection Path’ field.

### 7-5.8.6 Connection Path

The connection path of the data. For the "OutputExaN" form of this keyword, if a ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference is specified then the 'Connection Path' is the path specified in the ParamN, AssemN, AssemExaN, VariantN or VariantExaN reference.

### 7-5.8.7 Help String

The textual help string. When displayed, truncation may occur to meet the display capabilities of the configuration tool.

## 7-5.9 Examples

This section contains example sections of EDS files that specify the I/O characteristics of the device.

### 7-5.9.1 Poll & Strobe Capable Device, Poll is Default

```
[File]
...
[Device]
...
[IO_Info]
Default - 0x0001;          $ Poll
                           $ Bit mapped (0   None)
                           $ Bit 0 - Poll
                           $ Bit 1 = Strobe
PollInfo= 0x0003,          $ OK to Combine w/Poll or Strobe
    2,                      $ Default Input = Input2
    3;                      $ Default Output = Output3
StrobeInfo 0x0003,          $ OK to Combine w/Poll or Strobe
    1,                      $ Default Input = Input1
    4;                      $ Default Output = Output4

$   Input Connections
Input1
    1,                      $ 1 byte
    1,                      $ 1 bit is significant
    0x0003,                  $ Strobe or Poll Connection
    "Data",                  $ Name String
    6,                      $ Path Size
    "20 04 24 20 30 03",     $ Assy Obj Inst 20 Attr 3
    "This is a help string."; $ Help String

Input2=2, 0, 0x0003, "Status", 6, "20 04 24 21 30 03",
    "This is a help string.";

Input3 3, 0, 0x0003, "Data + Status", 6, "20 04 24 22 30 03",
    "This is a help string.";

$   Output Connections
Output1=
    1,                      $ 1 byte
    0,                      $ All bits are significant
```

```
0x0001,          $ Poll Only Connection
"Data",          $ Name String
6,              $ Path Size
"20 04 24 10 30 03", $ Assy Obj Inst 10 Attr 3
"This is a help string."; $ Help String

Output2 2, 0, 0x0001, "Reference", 6, "20 04 24 11 30 03",
  "This is a help string.";

Output3=3, 0, 0x0001, "Data + Reference",6, "20 04 24 12 30 03",
  "This is a help string.";

Output4=
  1,              $ Use Strobe Bit
  1,              $ 1 significant bit
  0x0002,         $ Strobe Only Connection
  "Data Bit",     $ Name String
  4,              $ Path Size
  "20 0E 24 01",  $ Sensor Obj Inst 1
  "This is a help string."; $ Help String

Output5
  1,              $ Use Strobe Bit
  1,              $ 1 significant bit
  0x0002,         $ Strobe Only Connection
  "Sync Bit",     $ Name String
  4,              $ Path Size
  "20 0E 24 02",  $ Sensor Obj Inst 2
  "This is a help string."; $ Help String

Output6=
  0,              $ Strobe Bit Ignored, Don't Map
  0,              $ All bits are significant (None)
  0x0002,         $ Strobe Only Connection
  "Sync Pulse",   $ Name String
  4,              $ Path Size
  "20 0E 24 03",  $ Sensor Obj Inst 3
  "This is a help string."; $ Help String

[ParamClass]
...
```



### **7-5.9.2 Strobe Only Device, Only One Input and One Output**

```
[File]
...
[Device]
...
[IO_Info]

Default = 0x0002;           $ Strobe Only

StrobeInfo=
  0x0002                    $ Strobe Only
  1;                        $ Default Input   Input1
  1;                        $ Default Output = Output1

$   Input Connections
Input1=
  1,                        $ 1 byte
  2,                        $ 2 bits are significant
  0x0002,                   $ Only Strobe Connection
  "Data & Status Bits",    $ Name String
  4,                        $ Path Size
  "20 0E 24 01",           $ Sensor Obj Inst 1
  "This is a help string."; $ Help String

$ -- Output Connections --
Output1=
  0,                        $ Strobe bit not used
  0,                        $ All bits are significant
  0x0002,                   $ Only Strobe Connection
  "Strobe Out Bit Not Used", $ Name String
  6,                        $ Path Size
  "20 0E 24 01",           $ Sensor Obj Inst 1
  "This is a help string."; $ Help String

[ParamClass]
...
```

### 7-5.9.3 Poll & Change of State Capable Device, Poll and Change of State is Default

```
[File]
...
[Device]
...
[IO_Info]

Default    0x0005;                $ Poll & Change of State
                                   $ Bit 0 = Poll
                                   $ Bit 2 =Change of State

PollInfo
    0x0005,                $ OK to Combine w/Poll or COS
    2,                    $ Default Input = Input2
    2;                    $ Default Output = Output2
COSInfo=
    0x0005,                $ OK to Combine w/Poll or COS
    1,                    $ Default Input = Input1
    1;                    $ Default Output = Output1

$ -- Input Connections --
Input1=
    1,                    $ 1 byte
    1,                    $ 1 bit is significant
    0x0005,                $ Poll or COS Connection
    "Data",                $ Name String
    6,                    $ Path Size
    "20 04 24 20 30 03",    $ Assy Obj Inst 20 Attr 3
    "This is a help string."; $ Help String

Input2=2, 0, 0x0005, "Status", 6, "20 04 24 21 30 03",
    "This is a help string.";

Input3=3, 0, 0x0005, "Data + Status", 6, "20 04 24 22 30 03",
    "This is a help string.";

$ -- Output Connections -
Output1=
    1,                    $ 1 byte
    0,                    $ All bits are significant
    0x0005,                $ Poll or COS Connection
    "Data",                $ Name String
    6,                    $ Path Size
    "20 04 24 10 30 03",    $ Assy Obj Inst 10 Attr 3
    "This is a help string."; $ Help String

Output2=2, 0, 0x0005, "Reference", 6, "20 04 24 11 30 03",
    "This is a help string.";

Output3=3, 0, 0x0005, "Data + Reference", 6, "20 04 24 12 30 03",
    "This is a help string.";

[ParamClass]
...
```

#### 7-5.9.4 Example Showing Use of Extended IO\_Info Keywords

```
[File]
...
[Device]
...
[IO_Info]
Default      0x0002;      $ Default IO Type mask: (Strobed)

PollInfo=    0x000F,      $ to combine (Poll, Strobe, COS, Cyclic)
              1,          $ Default Prod connection:  Input1
              0;          $ Default Cons connection:  (0: None)

PollInfoExa  0x000F,      $ to combine (Poll, Strobe, COS, Cyclic)
              1,          $ Default Prod connection: - InputExa1
              0;          $ Default Cons connection: - (0: None)

COSInfo=     0x000F,      $ to combine (Poll, Strobe, COS, Cyclic)
              1,          $ Default Prod connection: - Input1
              0;          $ Default Cons connection: - (0: None)

COSInfoExa=  0x000F,      $ to combine (Poll, Strobe, COS, Cyclic)
              2,          $ Default Prod connection:  InputExa2
              0;          $ Default Cons connection:  (0: None)

StrobeInfo   0x000F,      $ to combine (Poll, Strobe, COS, Cyclic)
              1,          $ Default Prod connection: - Input1
              0;          $ Default Cons connection: - (0: None)

StrobeInfoExa= 0x000F,    $ to combine (Poll, Strobe, COS, Cyclic)
              3,          $ Default Prod connection: - InputExa3
              0;          $ Default Cons connection: - (0: None)

CyclicInfo=   0x000F,      $ to combine (Poll, Strobe, COS, Cyclic)
              1,          $ Default Prod connection:  Input1
              0;          $ Default Cons connection:  (0: None)

CyclicInfoExa 0x000F,      $ to combine (Poll, Strobe, COS, Cyclic)
              4,          $ Default Prod connection: - InputExa4
              0;          $ Default Cons connection: - (0: None)

$          INPUT Connections: == Producing Connection Entries
Input1     1,              $ data size in bytes
           0,              $ num of significant bits (0: all)
           0x000F,         $ IO Type mask
           "Position Value", $ name for data (Instance)
           6,              $ link path size
           "20 04 24 01 30 03", $ link path (Assem | Instance | Attrib)
           "POSITION VALUE help string";

Input2     2,              $ data size in bytes
           0,              $ num of significant bits (0: all)
           0x000F,         $ IO Type mask
           "Position Value and Flags", $ name for data (Instance)
           6,              $ link path size
           "20 04 24 02 30 03", $ link path (Assem | Instance | Attrib)
           "POSITION VALUE and FLAGS help string";
```

```

Input3      2,                $ data size in bytes
            0,                $ num of significant bits (0: all)
            0x0003,           $ IO Type mask (not COS or cyclic )
            "Position Value and Velocity",$ name for data (Instance)
            6,                $ link path size
            "20 04 24 03 30 03", $ link path (Assem Instance | Attrib)
            "POSITION VALUE and VELOCITY help string";

Input4=      2,                $ data size in bytes
            0,                $ num of significant bits (0: all)
            0x000F,           $ IO Type mask
            "Position Value and Cam State", $ name for data (Instance)
            6,                $ link path size
            "20 04 24 04 30 03", $ link path (Assem Instance | Attrib)
            "POSITION VALUE and CAM STATE help string";

InputExa1    , $ data size in bytes, determined based on Assem size
            , $ num of significant bits, determined based on Assem definition
            0x000F,           $ IO Type mask
            ,                 $ name, determined based on Assem name
            ,                 $ link path size, not required
            VariantExa1,     $ link path, obtained from Assem
            ;                 $ help string, not required

InputExa2=   , $ data size in bytes, determined based on Assem size
            , $ num of significant bits, determined based on Assem definition
            0x000F,           $ IO Type mask
            ,                 $ name, determined based on Assem name
            ,                 $ link path size, not required
            VariantExa2,     $ link path, obtained from Assem
            ;                 $ help string, not required

InputExa3=   , $ data size in bytes, determined based on Assem size
            , $ num of significant bits, determined based on Assem definition
            0x000F,           $ IO Type mask
            ,                 $ name, determined based on Assem name
            ,                 $ link path size, not required
            VariantExa3,     $ link path, obtained from Assem
            ;                 $ help string, not required

InputExa4    , $ data size in bytes, determined based on Assem size
            , $ num of significant bits, determined based on Assem definition
            0x000F,           $ IO Type mask
            ,                 $ name, determined based on Assem name
            ,                 $ link path size, not required
            AssemExa1,       $ link path, obtained from Assem
            ;                 $ help string, not required

$ ===== Parameters Section =====
[Params]
    $ INPUT_Assembly SECTION: [111, 112, 113]
Param55-     $ Assembly Number: -- Polling mode
            0,
            6, "20 2F 24 01 30 6F", $(#111)
            0x0002,           $ descriptor { enumerated }
            0xC6, 1,          $ Type: USINT, size: 1 byte
            "Input Assy. (POLL)",
            "",

```

```

        "POLL selector help string",
        1, 4, 1,                      $ min, max, default
        0, 0, 0, 0,
        0, 0, 0, 0, 0 ;

Param56          $ Assembly Number: -- COS / Cyclic mode
0,
6, "20 2F 24 01 30 70",    $ (#112)
0x0002,              $ descriptor ( enumerated )
0xC6, 1,              $ Type: USINT, size: 1 byte
"Input Assem. (COS)",
"",
"COS selector help string",
1, 4, 1,                      $ min, max, default
0, 0, 0, 0,
0, 0, 0, 0, 0 ;

Param57          $ Assembly Number: -- BIT Strobed mode
0,
6, "20 2F 24 01 30 71",    $ (#113)
0x0002,              $ descriptor ( enumerated )
0xC6, 1,              $ Type: USINT, size: 1 byte
"Input Assem. (BIT)",
"",
"BIT selector help string",
1, 4, 1,                      $ min, max, default
0, 0, 0, 0,
0, 0, 0, 0, 0 ;

$ ----- Parameter Enumerating Section -----
[EnumPar]
Param55=          $ "Input Assembly (POLL)"
"Position Value",
"Position Value + Flag",
"Position Value + Velocity",
"Position Value + Cam State";

Param56=          $ "Input Assembly (COS )"
"Position Value",
"Position Value + Flag",
"Reserved",          $ not supported
"Position Value + Cam State";

Param57=          $ "Input Assembly (BIT )"
"Position Value",
"Position Value + Flag",
"Position Value + Velocity",
"Position Value + Cam State";

[Assembly]
VariantExal =      $ variant used for POLL assembly selection
"POLL Variant",,,,,
Param55,
0x01, Assem1,
0x02, Assem2,
0x03, Assem3,
0x04, Assem4;

```

```

VariantExa2          $ variant used for COS assembly selection
    "COS Variant",,,,,
    Param56,
    0x01, Assem1,
    0x02, Assem2,
    0x03, Assem5,
    0x04, Assem4;

VariantExa3 =        $ variant used for BIT strobed assembly selection
    "BIT Strobed Variant",,,,,
    Param57,
    0x01, Assem1,
    0x02, Assem2,
    0x03, Assem3,
    0x04, Assem4;

Assem1              "Position Value",          $ Position Assembly
    "20 04 24 01 30 03",
    1,
    0,
    ' ',
    8, Param1;          $ Position value

Assem2              "Position Value + Flag", $ Position and Flag Assembly
    "20 04 24 02 30 03",
    2,
    0,
    ' ',
    8, Param1,          $ Position value
    1, Param2,          $ Flag
    7, ;                $ undefined

Assem3 =            "Position Value + Velocity", $ Position and Flag Assembly
    "20 04 24 03 30 03",
    2,
    0,
    ' ',
    8, Param1,          $ Position value
    8, Param3;          $ Velocity

Assem4              "Position Value + Cam State", $ Position and Cam State Assembly
    "20 04 24 04 30 03",
    2,
    0,
    ' ',
    8, Param1,          $ Position value
    1, Param4,          $ Cam State
    7, ;                $ undefined

Assem5 =            "Reserved",                $ Reserved Assembly
    ,
    0,
    0,
    ' ',
    , ;                $ undefined

```

## 7-6 Variant I/O Info Section

The [Variant\_IO\_Info] section contains multiple sets of I/O characteristics of a device and is used when a device supports one of these sets of I/O characteristics based on the value of a specified parameter. If a [Variant\_IO\_Info] section exists in an EDS file a [IO\_Info] section shall not exist in the EDS file.

The entry keyword for all group selectors shall be “Selector”. The single field on this entry shall be a “symbolic”, the “symbolic” shall be one of AssemN, ProxyAssemN or ProxiedAssemN entry from the [Assembly] section or a ParamN, ProxyParamN or ProxiedParamN from the [Params] section. A reference to a “symbolic” may use the syntax “symbolic:N1:N2” where N1 specifies a bit offset into the “symbolic” and N2 specifies the number of bits to use. For example, “Assem1:0:4” indicates to use the first four bits of Assem1. The assembly or parameter specified as the “Selector” shall exist within the device and therefore shall have a path specified in its definition within the Assembly or Parameter section. The value of this field selects the specific set of I/O characteristics that are currently active for this device.

The “Selection” entry keyword shall be combined with a number (decimal). The single field on the “SelectionN” entry shall be a 16-bit unsigned integer (UINT). When the “SelectionN” field value is equal to the “Selector” value the [IO\_Info] section contents that follows shall be active.

The “SelectionN” section is terminated with the “EndSelectionN” keyword. The single field on the “EndSelectionN” entry shall be a 16-bit unsigned integer (UINT). The value of the single “EndSelectionN” field shall match the value of the preceeding “SelectionN” value field

The “Selector” entry keyword shall appear before the “SelectionN” entry keywords in the Variant I/O Info Section.

For a specific case and reducing variables – let’s say that we have only a “starter” type device (Identity attribute #2 is constant) – and attribute #100 of the motor control supervisor object will contain (and is settable) to one of three values. The EDS file, in this case, needs to expose the following [IO\_Info] statements:

**Figure 7-6.1 Example EDS Showing Variant IO Info Section Usage**

```
$ Given:
$   Device is a Motor Starter
$   {Identity Instance #1, Attribute #2 is the value 0x16}
$ - Vendor specific attribute #100 of the Control Supervisor (class 0x29)
$   publicizes the number of contactors in the device configuration

[Parameter]
  Param200 =,          $ Used as selector in [Variant_IO_Info]
    6, "20 29 24 01 30 64", $ Control Supervisor, attr #100
    0x0000,              $
    C6,                  $ USINT
    1,                   $ Data is 1 byte in size
    "Number of Contactors",
    "Count",
    "The number of motor control contactors in the device configuration.",
    0, 2, 1;
$
```

```

$
[Variant_IO_Info]
  Selector = Param200;      $ Base I/O Capability on configuration
$
$ When Control Supervisor Attribute #100 zero
$
  Selection1 = 0;          $ No contactor configuration
  Default=0x0001;
  PollInfo=0x0001,1,1;    $ Use Basic Overload Assemblies by default
  Input1=1,0,0x0001,
    "Basic Overload",6,"20 04 24 02 30 03",
    "Basic Overload";
  Output1=1,0,0x0001,
    "Motor Overload",6,"20 04 24 32 30 03",
    "Motor Overload";
  Output2=1,0,0x0001,
    "Extended Overload",6,"20 04 24 33 30 03",
    "Extended Overload";
  EndSelection1 = 0;
$
$ When Control Supervisor Attribute #100=one
$
  Selection2 1;          $ One contactor in use configuration
  Default 0x0001;
  PollInfo=0x0001,3,4;
  Input1=1,0,0x0001,
    "Basic Contactor",6,"20 04 24 01 30 03",
    "Basic Contactor";
  Input2 1,0,0x0001,
    "Basic Overload",6,"20 04 24 02 30 03",
    "Basic Overload";
  Input3=1,0,0x0001,
    "Basic Motor Starter",6,"20 04 24 02 30 03",
    "Basic Motor Starter";
  Output1 1,0,0x0001,
    "Motor Overload",6,"20 04 24 32 30 03",
    "Motor Overload";
  Output2=1,0,0x0001,
    "Extended Overload",6,"20 04 24 33 30 03",
    "Extended Overload";
  Output3 1,0,0x0001,
    "Basic Motor Starter",6,"20 04 24 34 30 03",
    "Basic Motor Starter";
  Output4=1,0,0x0001,
    "Extended Motor Starter 1",6,"20 04 24 35 30 03",
    "Extended Motor Starter 1";
  EndSelection2 1;
$
$ When Control Supervisor Attribute #100=two
$
  Selection3 = 2;
  Default 0x0001;
  PollInfo 0x0001,4,5;
  Input1=1,0,0x0001,
    "Basic Contactor",6,"20 04 24 01 30 03",
    "Basic Contactor";
  Input2=1,0,0x0001,
    "Basic Overload",6,"20 04 24 02 30 03",

```



```
"Basic Overload";
Input3=1,0,0x0001,
  "Basic Motor Starter",6,"20 04 24 02 30 03",
  "Basic Motor Starter";
Input4 1,0,0x0001,
  "Extended Contactor",6,"20 04 24 04
Output1=1,0,0x0001,
  "Motor Overload",6,"20 04 24 32 30 03",
  "Motor Overload";
Output2=1,0,0x0001,
  "Extended Overload",6,"20 04 24 33 30 03",
  "Extended Overload";
Output3=1,0,0x0001,
  "Basic Motor Starter",6,"20 04 24 34 30 03",
  "Basic Motor Starter";
Output4=1,0,0x0001,
  "Extended Motor Starter 1",6,"20 04 24 35 30 03",
  "Extended Motor Starter 1";
Output5=1,0,0x0001,
  "Extended Motor Starter 2",6,"20 04 24 36 30 03",
  "Extended Motor Starter 2";
EndSelection3 = 2;
```

## 7-7 Parameter Enumeration Strings Section

The parameter enumeration strings section begins with the keyword [EnumPar] and provides an enumeration list of parameter choices to present to the user. The entry keyword for all parameter enumeration strings consists of a combination of the string, “**Param**”, and the Parameter Object instance number (decimal) in the device, for example, “**Param1**”. Each parameter that supports enumerated strings must have an entry. The entries must be in ascending order

The fields in each entry contain the enumeration strings separated by commas, and a semicolon indicates the end of the entry. The position in the list defines the enumerated value assigned to the string. The first string in the enumerated list is assigned the minimum value of the parameter. Each ensuing string is assigned a value incremented by one. There must be one string for each integral value from the minimum to the maximum value of the parameter.

The table below shows the format of the parameter enumeration string section:

**Table 7-7.1 Parameter Enumeration String Section Format Table**

Field Name	Field Number	Data Type	Required/Optional
Enumeration String(s)	1 thru number of enumerations	ASCII Character Array	Required

**Figure 7-7.1 Sample Electronic Data Sheet with Param Enumerations**

```
$ Sample Electronic Data Sheet illustrating the Parameter Enumeration
$ Strings Section

[File]
...
[Device]
...
[IO_Info]
...
[ParamClass]
...
[Params]
...
[EnumPar]
  Param2 =          $ enums for param 2
                    "1 ms input delay ",
                    "10 ms input delay ",
                    "25 ms input delay";

  Param3 -          $ enums for param 3
                    "Light Operate",
                    "Dark Operate";

[Groups]
. . .
```

## 7-8 Internationalization Section

The Internationalization section is defined in Volume 1, Chapter 7. The DeviceNet specific keywords for the Internationalization section are defined here.

### 7-8.1 Input Entry Keywords

The **"InputN"** and **"InputExaN"** entry keyword is optional and shall contain the formatted fields shown in Table 7-8.1.

**Table 7-8.1 InputN and InputExaN Keyword Format**

Field name	Field Number	Data type	Required/Optional
International Name	1	STRINGI	Required
International Help String	2	STRINGI	Required

#### 7-8.1.1 International Name, Field 1

The international textual name. This field value overrides the corresponding **"InputN"** or **"InputExaN"** keyword name field value in the [IO\_Info] section.

#### 7-8.1.2 International Help String, Field 2

The international textual help string. This field value overrides the corresponding **"InputN"** or **"InputExaN"** keyword help string field value in the [IO\_Info] section.

### 7-8.2 Output Entry Keywords

The **"OutputN"** and **"OutputExaN"** entry keyword is optional and shall contain the formatted fields shown in Table 7-8.2.

**Table 7-8.2 OutputN and OutputExaN Keyword Format**

Field name	Field Number	Data type	Required/Optional
International Name	1	STRINGI	Required
International Help String	2	STRINGI	Required

#### 7-8.2.1 International Name, Field 1

The international textual name. This field value overrides the corresponding **"OutputN"** or **"OutputExaN"** keyword name field value in the [IO\_Info] section.

#### 7-8.2.2 International Help String, Field 2

The international textual help string. This field value overrides the corresponding **"OutputN"** or **"OutputExaN"** keyword help string field value in the [IO\_Info] section.

### 7-8.3 Selection and EndSelection Entry Keywords

To support internationalization of the [Variant\_IO\_Info] section **"Selection"** and **"EndSelection"** keywords shall be supported in the [Internationalization] section.

The **"Selection"** and **"EndSelection"** entry keywords correspond to the **"Selection"** and **"EndSelection"** entry keywords in the [Variant\_IO\_Info] section (for example, Selection1 in the [Internationalization] section corresponds to Selection1 in the [Variant\_IO\_Info] section).

The **"Selection"** entry keyword shall be combined with a number (decimal). The **"SelectionN"** entry keyword is optional and contains no fields (for example Selection1 = ;).

The **"EndSelection"** entry keyword shall be combined with a number (decimal). The **"EndSelectionN"** entry keyword is conditional and contains no fields (for example EndSelection1 = ;). The **"EndSelectionN"** keyword is required if the **"SelectionN"** keyword exists and not allowed if the **"SelectionN"** keyword does not exist.

The **"SelectionN"** and **"EndSelectionN"** keywords shall be specified as a pair with N matching (for example, Selection1 paired with EndSelection1).

The valid keywords between a **"SelectionN"** and **"EndSelectionN"** keyword are **"InputN"**, **"InputExaN"**, **"OutputN"** and **"OutputExaN"**. The definition of the **"InputN"**, **"InputExaN"**, **"OutputN"**, **"OutputExaN"** keywords between **"SelectionN"** and **"EndSelectionN"** keywords is specified in Chapter 7-8.1 and 7-8.2.

The **"InputN"**, **"InputExaN"**, **"OutputN"**, **"OutputExaN"** keywords in the [Internationalization] section corresponds to the same keyword bracketed by **"SelectionN"** and **"EndSelectionN"** in the [Variant\_IO\_Info] section.

Figure 7-8.1 is an EDS section that shows an example of internationalizing a Variant\_IO\_Info section. The [Variant\_IO\_Info] section being translated is defined in Figure 7-6.1.

**Figure 7-8.1 Example EDS Showing Internationalization of Variant\_IO\_Info Section**

```
$ Refer to Figure 7-7.1 for the [Variant_IO_Info] section definition
[Internationalization]
...

$ When Control Supervisor Attribute #100=zero
$
Selection1    0;          $ No contactor configuration

Input1=
{ 3,
  {"eng",0xD0,4,"Basic Overload"},
  {"spa",0xD0,4,"Sobrecarga Básica"},          $ Spanish
  {"deu",0xD0,4,"Grundlegende Überlastung"}    $ German
},
{ 3,
  {"eng",0xD0,4,"Basic Overload"},
  {"spa",0xD0,4,"Sobrecarga Básica"},          $ Spanish
  {"deu",0xD0,4,"Grundlegende Überlastung"}    $ German
};
```

```

Output1
{ 3,
{"eng",0xD0,4,"Motor Overload"},
{"spa",0xD0,4,"Sobrecarga Del Motor"},          $ Spanish
{"deu",0xD0,4,"Bewegungsüberlastung"}          $ German
},
{ 3,
{"eng",0xD0,4,"Motor Overload"},
{"spa",0xD0,4,"Sobrecarga Del Motor"},          $ Spanish
{"deu",0xD0,4,"Bewegungsüberlastung"}          $ German
};

Output2
{ 3,
{"eng",0xD0,4,"Extended Overload"},
{"spa",0xD0,4,"Sobrecarga Extendida"},          $ Spanish
{"deu",0xD0,4,"Ausgedehnte Überlastung"}        $ German
},
{ 3,
{"eng",0xD0,4,"Extended Overload"},
{"spa",0xD0,4,"Sobrecarga Extendida"},          $ Spanish
{"deu",0xD0,4,"Ausgedehnte Überlastung"}        $ German
};
EndSelection1 0;

$ When Control Supervisor Attribute #100=one
$
Selection2 = 1;          $ One contactor in use configuration

Input1
{ 3,
{"eng",0xD0,4,"Basic Contactor"},
{"spa",0xD0,4,"Contactor Básico"},              $ Spanish
{"deu",0xD0,4,"Grundlegender Kontaktgeber"}     $ German
},
{ 3,
{"eng",0xD0,4,"Basic Contactor"},
{"spa",0xD0,4,"Contactor Básico"},              $ Spanish
{"deu",0xD0,4,"Grundlegender Kontaktgeber"}     $ German
};

Input2
{ 3,
{"eng",0xD0,4,"Basic Overload"},
{"spa",0xD0,4,"Sobrecarga Básica"},              $ Spanish
{"deu",0xD0,4,"Grundlegende Überlastung"}        $ German
},
{ 3,
{"eng",0xD0,4,"Basic Overload"},
{"spa",0xD0,4,"Sobrecarga Básica"},              $ Spanish
{"deu",0xD0,4,"Grundlegende Überlastung"}        $ German
};

Input3
{ 3,
{"eng",0xD0,4,"Basic Motor Starter"},
{"spa",0xD0,4,"Arrancador Básico Del Motor "},   $ Spanish
{"deu",0xD0,4,"Grundlegender Bewegungsstarter"} $ German
}

```

```

},
{ 3,
{"eng",0xD0,4,"Basic Motor Starter"},
{"spa",0xD0,4,"Arrancador Básico Del Motor"},      $ Spanish
{"deu",0xD0,4,"Grundlegender Bewegungsstarter"}    $ German
};

Output1=
{ 3,
{"eng",0xD0,4,"Motor Overload"},
{"spa",0xD0,4,"Sobrecarga Del Motor"},              $ Spanish
{"deu",0xD0,4,"Bewegungsüberlastung"}              $ German
},
{ 3,
{"eng",0xD0,4," Motor Overload"},
{"spa",0xD0,4,"Sobrecarga Del Motor"},              $ Spanish
{"deu",0xD0,4,"Bewegungsüberlastung"}              $ German
};

Output2=
{ 3,
{"eng",0xD0,4,"Extended Overload"},
{"spa",0xD0,4,"Sobrecarga Extendida"},              $ Spanish
{"deu",0xD0,4,"Ausgedehnte Überlastung"}            $ German
},
{ 3,
{"eng",0xD0,4,"Extended Overload"},
{"spa",0xD0,4,"Sobrecarga Extendida"},              $ Spanish
{"deu",0xD0,4,"Ausgedehnte Überlastung"}            $ German
};

Output3=
{ 3,
{"eng",0xD0,4,"Basic Motor Starter"},
{"spa",0xD0,4,"Arrancador Básico Del Motor"},      $ Spanish
{"deu",0xD0,4,"Grundlegender Bewegungsstarter"}    $ German
},
{ 3,
{"eng",0xD0,4,"Basic Motor Starter "},
{"spa",0xD0,4,"Arrancador Básico Del Motor"},      $ Spanish
{"deu",0xD0,4,"Grundlegender Bewegungsstarter"}    $ German
};

Output4=
{ 3,
{"eng",0xD0,4,"Extended Motor Starter 1"},
{"spa",0xD0,4,"Arrancador Extendido 1 Del Motor"}, $ Spanish
{"deu",0xD0,4,"Ausgedehnter Bewegungsstarter 1"}  $ German
},
{ 3,
{"eng",0xD0,4," Extended Motor Starter 1"},
{"spa",0xD0,4,"Arrancador Extendido 1 Del Motor"}, $ Spanish
{"deu",0xD0,4,"Ausgedehnter Bewegungsstarter 1"}  $ German
};

EndSelection2 = 1;

$ When Control Supervisor Attribute #100=two

```

```

$
Selection3 = 2;

Input1=
{ 3,
{"eng",0xD0,4,"Basic Contactor"},
{"spa",0xD0,4,"Contactor Básico"},           $ Spanish
{"deu",0xD0,4,"Grundlegender Kontaktgeber"}   $ German
},
{ 3,
{"eng",0xD0,4,"Basic Contactor"},
{"spa",0xD0,4,"Contactor Básico"},           $ Spanish
{"deu",0xD0,4,"Grundlegender Kontaktgeber"}   $ German
};

Input2=
{ 3,
{"eng",0xD0,4,"Basic Overload"},
{"spa",0xD0,4,"Sobrecarga Básica"},           $ Spanish
{"deu",0xD0,4,"Grundlegende Überlastung"}     $ German
},
{ 3,
{"eng",0xD0,4,"Basic Overload"},
{"spa",0xD0,4,"Sobrecarga Básica"},           $ Spanish
{"deu",0xD0,4,"Grundlegende Überlastung"}     $ German
};

Input3=
{ 3,
{"eng",0xD0,4,"Basic Motor Starter"},
{"spa",0xD0,4,"Arrancador Básico Del Motor"}, $ Spanish
{"deu",0xD0,4,"Grundlegender Bewegungsstarter"} $ German
},
{ 3,
{"eng",0xD0,4,"Basic Motor Starter"},
{"spa",0xD0,4,"Arrancador Básico Del Motor"}, $ Spanish
{"deu",0xD0,4,"Grundlegender Bewegungsstarter"} $ German
};

Input4
{ 3,
{"eng",0xD0,4,"Extended Contactor"},
{"spa",0xD0,4,"Contactor Extendido"},           $ Spanish
{"deu",0xD0,4,"Ausgedehnter Kontaktgeber"}     $ German
},
{ 3,
{"eng",0xD0,4,"Extended Contactor"},
{"spa",0xD0,4,"Contactor Extendido"},           $ Spanish
{"deu",0xD0,4,"Ausgedehnter Kontaktgeber"}     $ German
};

Output1
{ 3,
{"eng",0xD0,4,"Motor Overload"},
{"spa",0xD0,4,"Sobrecarga Del Motor"},           $ Spanish
{"deu",0xD0,4,"Bewegungsüberlastung"}           $ German
},
{ 3,

```

```

{"eng",0xD0,4,"Motor Overload"},
{"spa",0xD0,4,"Sobrecarga Del Motor"},           $ Spanish
{"deu",0xD0,4,"Bewegungsüberlastung"}           $ German
};

Output2
{ 3,
{"eng",0xD0,4,"Extended Overload"},
{"spa",0xD0,4,"Sobrecarga Extendida"},           $ Spanish
{"deu",0xD0,4,"Ausgedehnte Überlastung"}         $ German
},
{ 3,
{"eng",0xD0,4,"Extended Overload"},
{"spa",0xD0,4,"Sobrecarga Extendida"},           $ Spanish
{"deu",0xD0,4,"Ausgedehnte Überlastung"}         $ German
};

Output3
{ 3,
{"eng",0xD0,4,"Basic Motor Starter"},
{"spa",0xD0,4,"Arrancador Básico Del Motor"},     $ Spanish
{"deu",0xD0,4,"Grundlegender Bewegungsstarter"}  $ German
},
{ 3,
{"eng",0xD0,4,"Basic Motor Starter"},
{"spa",0xD0,4,"Arrancador Básico Del Motor"},     $ Spanish
{"deu",0xD0,4,"Grundlegender Bewegungsstarter"}  $ German
};

Output4
{ 3,
{"eng",0xD0,4,"Extended Motor Starter 1"},
{"spa",0xD0,4,"Arrancador Extendido 1 Del Motor"}, $ Spanish
{"deu",0xD0,4,"Ausgedehnter Bewegungsstarter 1"} $ German
},
{ 3,
{"eng",0xD0,4,"Extended Motor Starter 1"},
{"spa",0xD0,4,"Arrancador Extendido 1 Del Motor"}, $ Spanish
{"deu",0xD0,4,"Ausgedehnter Bewegungsstarter"}   $ German
};

Output5 "Extended Motor Starter 2","Extended Motor Starter 2";
{ 3,
{"eng",0xD0,4,"Extended Motor Starter 2"},
{"spa",0xD0,4,"Arrancador Extendido 2 Del Motor"}, $ Spanish
{"deu",0xD0,4,"Ausgedehnter Bewegungsstarter 2"} $ German
},
{ 3,
{"eng",0xD0,4,"Extended Motor Starter 2"},
{"spa",0xD0,4,"Arrancador Extendido 2 Del Motor"}, $ Spanish
{"deu",0xD0,4,"Ausgedehnter Bewegungsstarter 2"} $ German
};

EndSelection3 = 2;

```



This page is intentionally left blank

## **Volume 3: DeviceNet Adaptation of CIP**

# **Chapter 8: Physical Layer**

---

## Contents

8-1	Introduction.....	3
8-1.1	Components of the Physical Layer.....	3
8-1.2	Physical Layer and Media Features .....	4
8-1.3	Physical Signaling.....	4
8-2	Physical Layer.....	5
8-2.1	General Physical Layer Requirements .....	5
8-2.2	Device Net Physical Layer Schematics.....	7
8-3	Transmission Media.....	8
8-3.1	Topology .....	9
8-3.2	Terminating Resistors .....	10
8-3.3	Connectors .....	10
8-3.4	Device Taps .....	11
8-3.5	Power Taps.....	11
8-3.6	Network Grounding .....	13
8-3.7	Temporary Terminal Support.....	15
8-3.8	Media .....	17
8-3.9	DeviceNet Tap Specifications.....	57
8-3.10	Power Tap Profiles.....	75
8-3.11	DeviceNet Power Specifications.....	78
8-3.12	Connectors .....	82
8-3.13	Transceiver.....	102
8-3.14	Non-isolated Physical Layer .....	104
8-3.15	Isolated Physical Layer .....	105
8-4	Powered Node Implementation .....	107
8-4.1	Node Power Regulator.....	107
8-4.2	Use of Regulator in Power Design.....	108
8-5	Configuring Network Power.....	113
8-5.1	Defining Your Power Configuration .....	113
8-5.2	Load Limits.....	119
8-5.3	System Tolerance .....	119
8-5.4	Avoiding Errors.....	120
8-5.5	Power Supply Options .....	120

## **8-1 Introduction**

This chapter outlines the *minimum* electronic requirements for products connecting to DeviceNet. Included are instructions for making physical connections, providing power, and applying CAN signaling to the DeviceNet Physical Layer and Media.

The Physical Layer of DeviceNet is described in section 8-1.1. The DeviceNet Media is described in section 8-1.2.

### **8-1.1 Components of the Physical Layer**

The DeviceNet Physical Layer contains two components, the Medium Attachment Unit, and the Transmission Media. Within the DeviceNet Specification, the term Physical Layer will be used to discuss the elements of the Medium Attachment Unit. This includes the driver/receiver circuitry and other circuitry used to connect the node to the Transmission Medium, called the Physical Medium Attachment in the ISO model. Also included in the Physical Layer is the definition of the electrical and mechanical interface to the Transmission Medium, called the Media Dependent Interface in the ISO model.

### 8-1.2 Physical Layer and Media Features

The DeviceNet Physical Layer and Media are designed to include the following features:

- Use of CAN technology
- Small size and low cost
- Linear bus topology
- Ability to operate at three data rates:
  - 125 K baud up to 500 m maximum
  - 250 K baud up to 250 m maximum
  - 500K baud up to 100 m maximum
- Various media containing both signal and power conductors
- Low loss, low delay cable
- Support of various media for drop line or trunk line
- Support of drop lines as long as 6 m/20 feet
- Support of as many as 64 nodes
- Node removal without severing the network
- Ability to support both isolated and non-isolated Physical Layers simultaneously
- Support of sealed media
- Protection from wiring errors

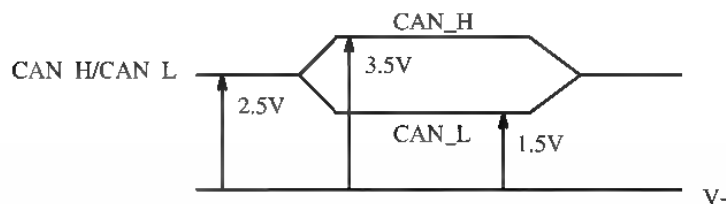
### 8-1.3 Physical Signaling

The BOSCH CAN specification defines two complimentary logical levels: 'dominant' and 'recessive'. During simultaneous transmission of 'dominant' and 'recessive' bits, the resulting bus value will be 'dominant'. For example, in case of a wired-AND implementation of the bus (as with DeviceNet), the 'dominant' level would be represented by a logical '0' and the 'recessive' level by a logical '1'.

Physical states (e.g. electrical voltage) that represent the logical levels are not given in the CAN specification. The specification used for these levels is contained in the ISO 11898 standard. Typically, for a node disconnected from the bus, the recessive (high impedance) levels for CAN\_L and CAN\_H are 2.5 volts (0 volts differential).

The typical dominant (low impedance driven) levels are 1.5 volts for CAN\_L and 3.5 volts for CAN\_H (2 volts differential) as shown in Figure 8-1.1.

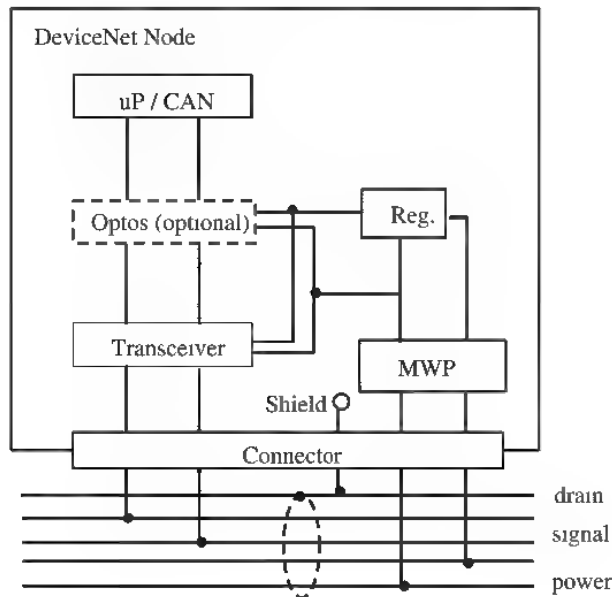
Figure 8-1.1 CAN\_H and CAN\_L Signal Levels



## 8-2 Physical Layer

The Physical Layer is comprised of the transceiver, connector, mis-wiring protection circuitry, regulator, and optional optical isolation. Figure 8-2.1 is a block diagram of the different parts of the Physical Layer. In this section, the transceiver, mis-wiring protection (MWP) example and optional isolation will be discussed. The connector is discussed in section 8-3.12. The regulator is discussed in section 8-4.

**Figure 8-2.1 Physical Layer Block Diagram**



### 8-2.1 General Physical Layer Requirements

Examples in this section show how to use a typical integrated transceiver in a DeviceNet product. Not all transceivers are the same, so care should be taken to make sure that any transceiver used allows the device to meet the specification given below for the DeviceNet physical layer.

- General Physical Layer Requirements
- Transmitter Requirements
- Receiver Requirements

**Table 8-2.1 General Physical Layer Requirements**

General Characteristic	Specification
Bit rates	125K, 250K, 500K
Distance with thick trunk	500 m at 125 Kbaud 250 m at 250 Kbaud 100 m at 500 Kbaud
Number of nodes	64
Signaling	CAN
Modulation	baseband
Encoding	NRZ with bit stuffing
Media Coupling	DC coupled differential Tx/Rx
Isolation <sup>1</sup> (Current Limit) Between network conductors and earth ground and between network conductors and any other conductors common between nodes. (Note: Network Conductors are V-, V+, CAN HI and CAN LO)	+/- 500 Volt DC test voltage – maximum current flow = 1 milliamp (optional opto-isolators on node side of transceiver)
Differential input impedance (recessive state) <sup>2</sup>	Shunt C = 24 pF maximum plus 12 pF/ft maximum of permanently attached drop line. Capacitance values tested @ 500KHz Shunt R = 20 K Ohms min.
Absolute max. voltage range	-25 to +18 Volts (CAN_H, CAN_L) <sup>3</sup>
<ol style="list-style-type: none"> <li>1 This specification is intended only to prevent large loops, which can contribute to signal interference. Meeting this requirement does not imply meeting any particular safety requirement. The vendor is responsible for determining applicable safety standards and assuring compliance to those standards</li> <li>2 The allocation of 12pF/ft shall be used for only permanent attached drop line cable. This allocation is strictly for the physical length of the wire/cable connected to the transceiver.</li> <li>3 Voltages at CAN_H and CAN_L are referenced to the transceiver IC ground pin. This voltage will be higher than the V- terminal by an amount equal to the voltage drop on the Schottky diode. This voltage should be 0.6 V maximum.</li> </ol>	

**Table 8-2.2 Transmitter Requirements**

Transmitter Characteristic	Specification
Differential Output level (nominal)	2.0 Volts p-p
Differential Output level (minimum) (@ connector, 50 Ohms load)	1.5 Volts p-p
Minimum Recessive Bus voltage @ CAN_H and CAN_L	2.0 Volts <sup>1</sup>
Maximum Recessive Bus voltage @ CAN_H and CAN_L	3.0 Volts <sup>1</sup>
Transmitter delay	120 ns max opto(40)+xcvr(80)
Output short circuit protection	internally limited
<ol style="list-style-type: none"> <li>1 Voltages at CAN_H and CAN_L are referenced to the transceiver IC ground pin. This voltage will be higher than the V- terminal by an amount equal to the voltage drop on the Schottky diode. This voltage should be 0.6 V maximum.</li> </ol>	

**Table 8-2.3 Receiver Requirements**

Receiver Characteristic	Specification
Differential Input Voltage Dominant	0.95 Volts min
Differential Input Voltage Recessive	0.45 Volts max
Hysteresis	150 mV typical
Receiver delay	130 ns max opto(40)+xcvr(90)
Operating voltage range	-5 to +10 Volts (CAN_H, CAN_L) <sup>1</sup>

<sup>1</sup> Voltages at CAN\_H and CAN\_L are referenced to the transceiver IC ground pin. This voltage will be higher than the V- terminal by an amount equal to the voltage drop on the Schottky diode. This voltage should be 0.6 V maximum.

## 8-2.2 Device Net Physical Layer Schematics

**Figure 8-2.2 DeviceNet Integrated Transceiver**

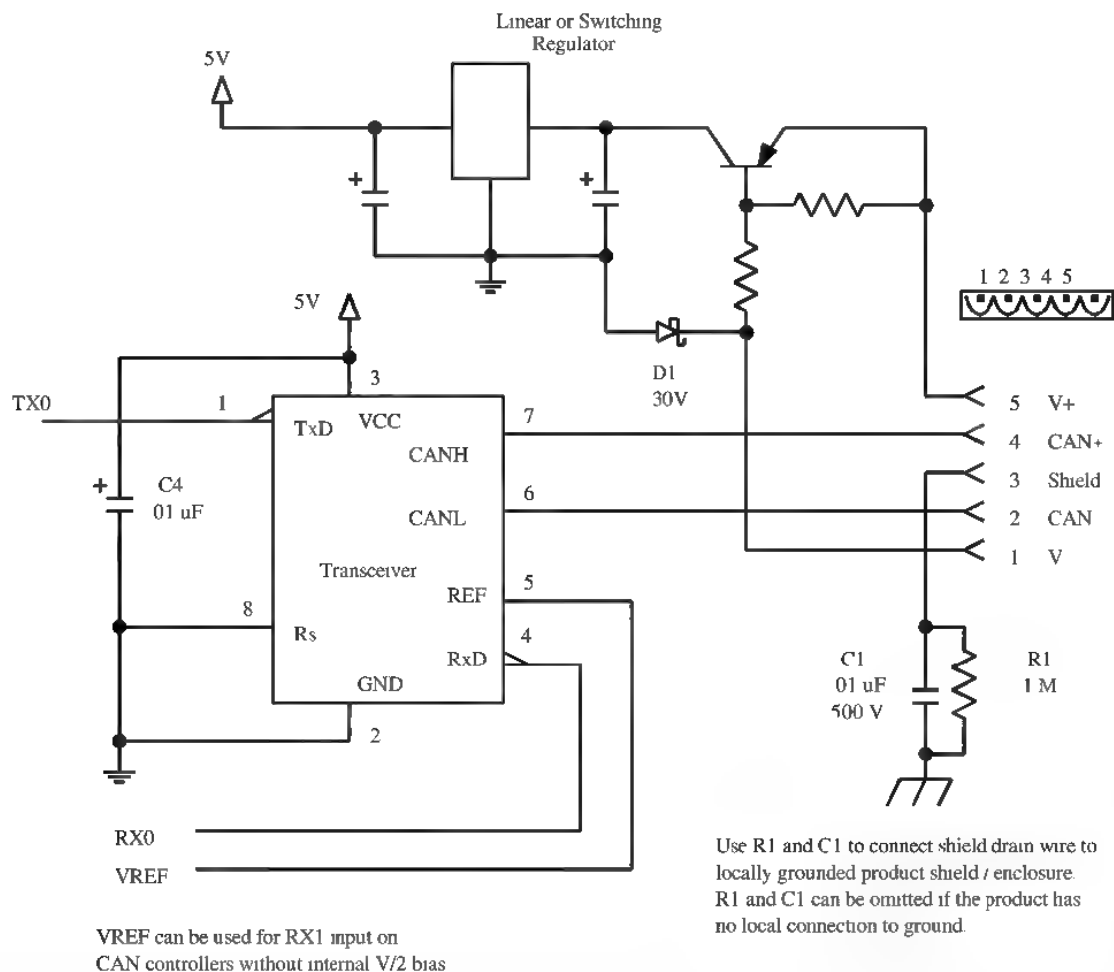
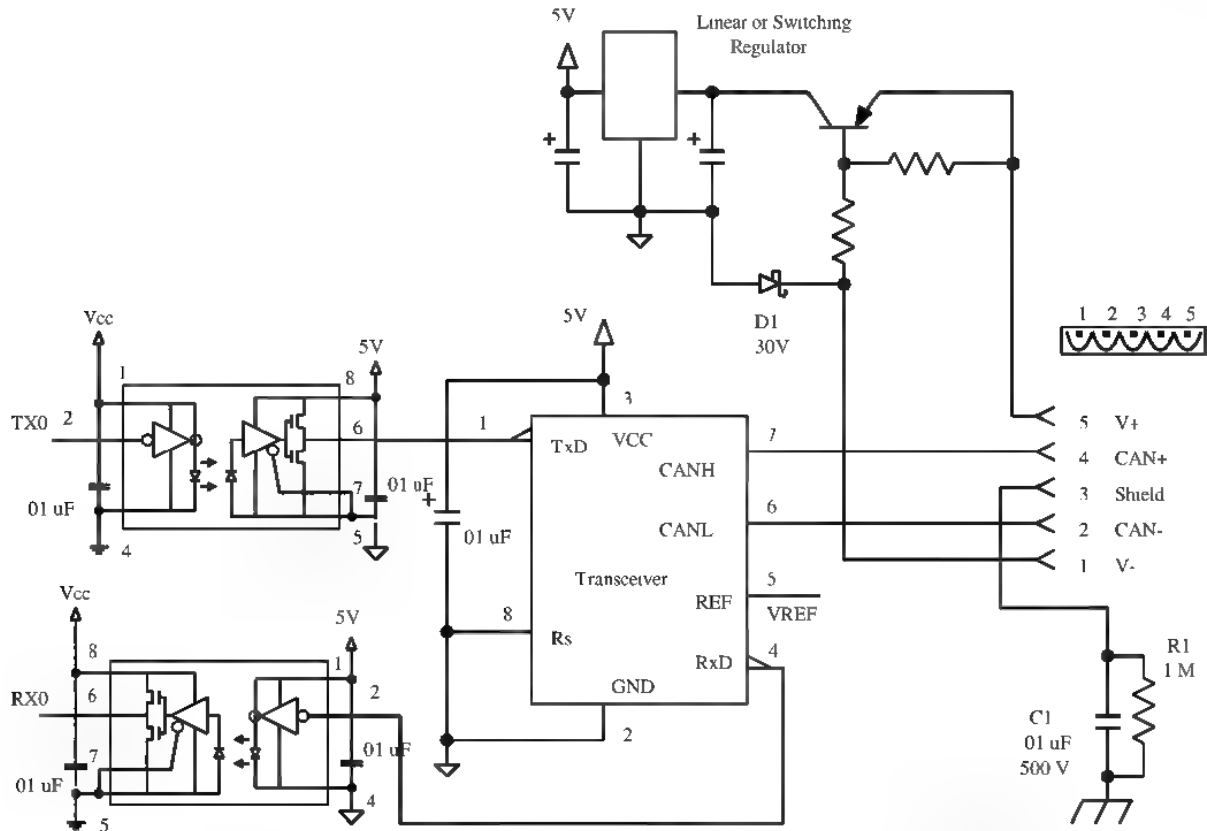




Figure 8-2.3 DeviceNet Isolated Integrated Transceiver



The RX1 input on the CAN controller must be tied to a  $V_{dd}/2$  voltage source referenced to the ground on the CAN controller side of the opto couplers.

Use R1 and C1 to connect shield drain wire to locally grounded product shield / enclosure. R1 and C1 can be omitted if the product has no local connection to ground

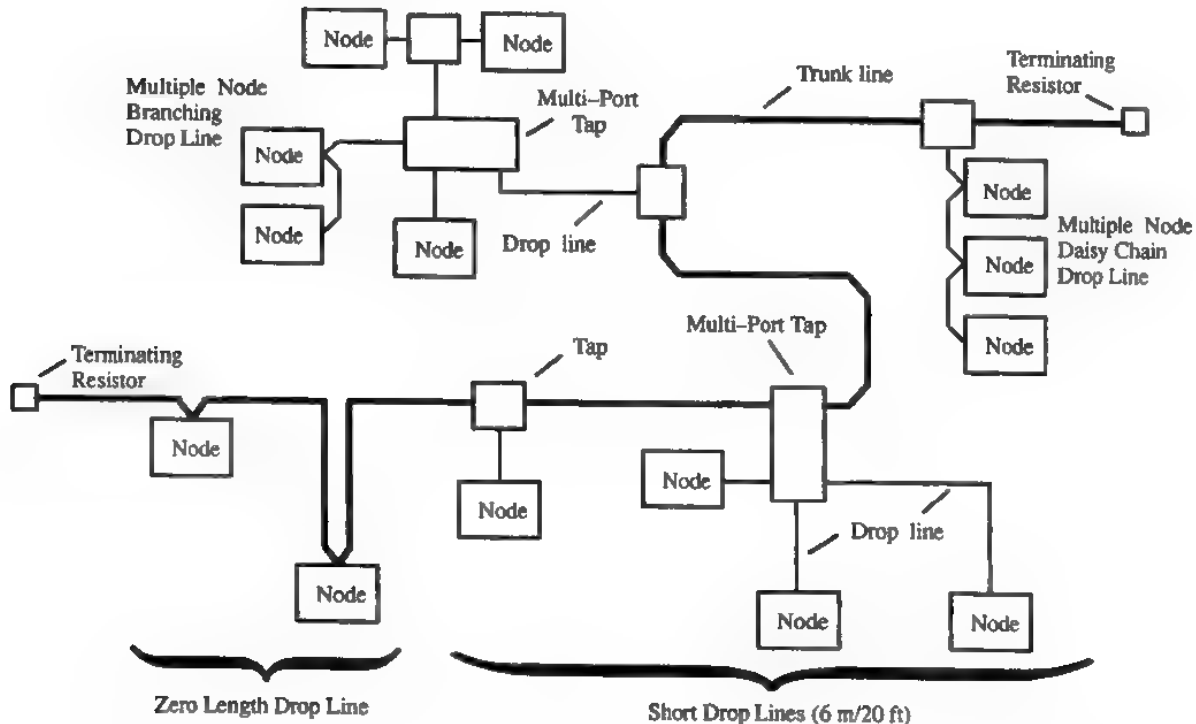
### 8-3 Transmission Media

The following sections describe the characteristics of the Transmission Media for DeviceNet. There are two major cable types round shielded and flat unshielded. Heavier cables allow longer trunk line distances and more sturdy trunk lines. Thinner Cables provides easier routing and termination of either trunk lines or drop lines.

### 8-3.1 Topology

The DeviceNet media has a linear bus topology. Terminating resistors are required on each end of the trunk line. Drop lines as long as 6 m (20 feet) each are permitted, allowing one or more nodes to be attached. DeviceNet allows branching structures only on the drop line. For information about the power delivery capability on the trunk line and drop line refer to section 8-5.

Figure 8-3.1 DeviceNet Media Topology



The total amount of trunk line allowable on the network depends upon the data rate and the type of cable used. The cable distance between any two points in the cable system must not exceed the *Maximum Cable Distance* allowed for the baud rate. For trunk lines constructed of only one type of cable, refer to cable profiles in section 8-3.8 to determine the Maximum Cable Distance based on the data rate and the type of cable used. Cable distance between two points includes *both* trunk line cable length and drop line cable length that exists between the two points.

DeviceNet allows mixing of different types of cables in a trunk system. The respective cable types in section 8-3.8 detail the equivalencies when mixing different types of cables in trunk lines.

Drop line length is the longest cable distance of those measured from the tap on the trunk line to each of the transceivers of the nodes on the drop line. This distance includes any dropline cable, which might be permanently attached to the device. The total amount of drop line allowable on the network depends upon the data rate. Refer to the cable profile in section 8-3.8 when determining the number and length of drop lines.

### 8-3.2 Terminating Resistors

DeviceNet requires a terminating resistor to be installed at each end of the trunk. The resistor requirements are:

- 121 ohm
- 1 % Metal Film
- 1/4 Watt

**Important:** Terminating resistors should never be included in nodes. Inclusion of this capability could easily lead to a network with improper termination (too high or too low an impedance) potentially causing failure. For example removal of a node, which includes a terminating resistor, could result in network failure.

**Important:** Terminating resistors should not be installed at the end of a drop line, only at the two ends of the trunk line.

### 8-3.3 Connectors

All connectors must support five conductors, which accommodate a signal pair, power pair, and a drain wire<sup>1</sup>. See section 8-3.12 for pin outs for all DeviceNet connectors.

It is recommended that the connector should be keyed such that a DeviceNet cable exits the instrument or device without interfering with any indicators, auxiliary connectors or anything else that may require access in the field. The mating DeviceNet receptacle on the instrument or device should be mounted such that the key orientation allows the cable to have no interference with indicators, auxiliary connectors, or anything else that may require access in the field. See figures 8.3-31 and 8.3-33 for details.

Hard-wired connections, such as direct soldering, crimping, screw terminal blocks and barrier strips, are allowed. However, these methods must support the requirement that the node is removable without severing the trunk.

Any DeviceNet connector shall meet the following minimum requirements:

- All wiping contacts must be gold plated.
- A minimal operating voltage of 25V.
- A nominal contact resistance of less than 1 mOhm and less than 5 mOhm over life.

**Important:** All nodes attaching to DeviceNet with a connector must have male pins. This applies to sealed and unsealed connectors and all nodes whether they are consuming or providing power.

**Important:** Whatever connector solution is chosen it is mandatory that devices can be removed without severing or disturbing the network.

**Important:** Wires should not be installed while the network is active. This will prevent problems such as shorting the network supply or disrupting communications.

---

<sup>1</sup> Connectors for use in only flat trunk systems can be made to support only four wires, since the drain wire is not used in this type of physical media.

### 8-3.4 Device Taps

Device taps provide points of attachment onto the trunk line. Devices can be connected to the network either directly to the tap or with drop line. Taps also provide easy removal of a device without disrupting network operation. For detailed specifications, refer to section 8-3.9.

### 8-3.5 Power Taps

A power tap connects the power supply to the trunk line. Power taps differ from device taps in that they can contain the following:

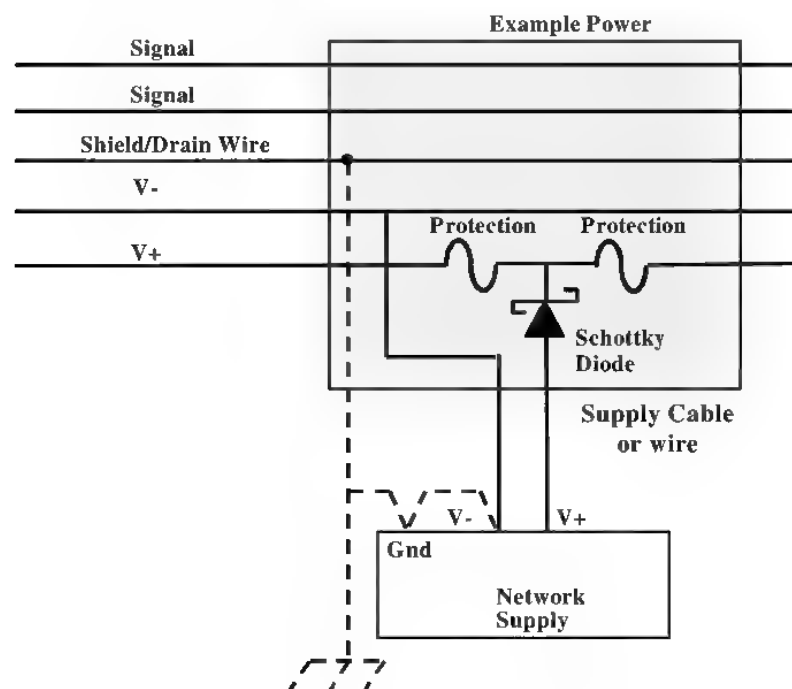
- A *Schottky* Diode which connects to the power supply V+ and allows for multiple supplies to be connected (this eliminates the need for custom power supplies). See section 8-3.11.4, Schottky Diode Specification, for more information about the Schottky Diode.
- Two fuses or circuit breakers to protect the bus from excess current, which could damage the cable and connectors.

When connected to the network, a power tap may provide the following:

- a continuous connection for the signal, drain and V- wires
- current limiting in each direction from the tap
- a connection to the shield/drain wire for grounding the network

The following figure illustrates the components of a DeviceNet power tap. Power Taps may be simplified for systems using single power supplies. For more detailed specifications on power taps refer to section 8-3.10.

Figure 8-3.2 Diagram of DeviceNet Power Tap



Power taps for DeviceNet, like the one shown in Figure 8-3.2, should have the following characteristics:

- Specified ratings for power supply and network currents
- Schottky Diode connected to the power supply for use on circuits with more than one supply.
- Fuses or circuit breakers to limit current on the bus to the specified current levels in each direction from the tap. These are to be used if current limiting in the power supply is insufficient.
- It is permissible to increase the length of the cable between the power supply and power tap for purposes of remote locating the power supply. The cable length from the power supply to the power tap shall conform to the wire gauge versus length as detailed in Figure 8-3.3, Figure 8-3.4 and Table 8-3.1. Note: Wire gauge sizes greater than AWG 10 (1.65 mm<sup>2</sup>) may not be compatible with some DeviceNet connectors.

Figure 8-3.3 Power Supply Cable Length Versus AWG (Feet)

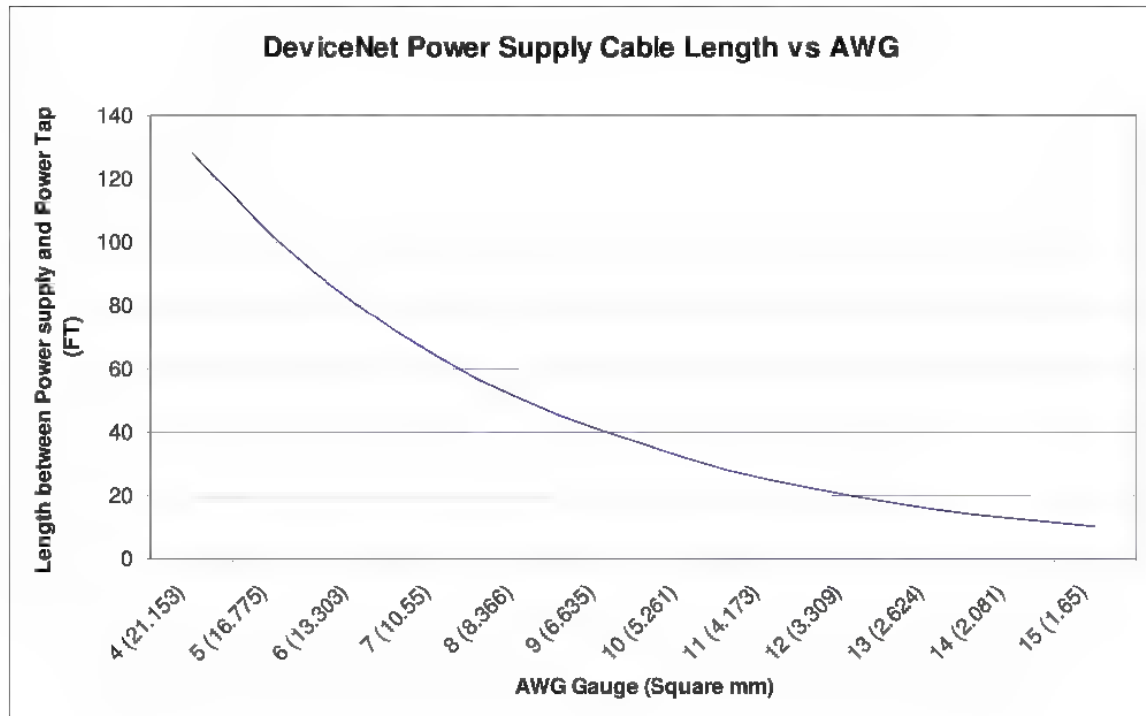


Figure 8-3.4 Power Supply Cable Length Versus AWG (Meters)

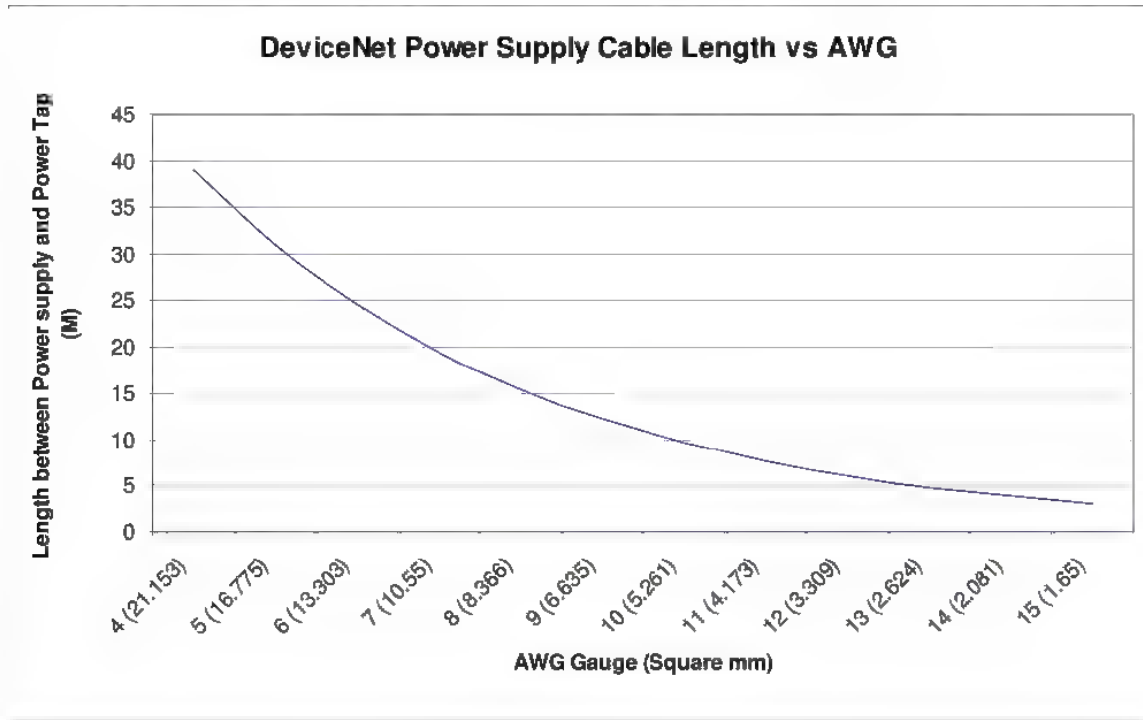


Table 8-3.1 Cable Gauge and Length

Gauge (AWG)	AWG (mm <sup>2</sup> )	Length (ft)	Length (mm)
15	15 (1.65)	10.00	3.05
14	14 (2.08)	12.59	3.84
13	13 (2.62)	15.88	4.84
12	12 (3.31)	20.03	6.10
11	11 (4.17)	25.24	7.69
10	10 (5.26)	31.84	9.70
9	9 (6.63)	40.15	12.24
8	8 (8.37)	50.62	15.43
7	7 (10.56)	63.83	19.46
6	6 (13.30)	80.49	24.53
5	5 (16.77)	101.50	30.94
4	4 (21.15)	127.97	39.00

### 8-3.6 Network Grounding

DeviceNet should be grounded at ONE location. Grounding at more than one location may produce ground loops, while not grounding the network will increase sensitivity to ESD and outside noise sources. The single grounding location should be at a power tap. Sealed DeviceNet power taps are designed to accommodate grounding. Grounding near the physical center of the network is also desired.

The trunk drain/shield should be attached to the power supply ground or V- with a copper conductor, that is either solid, stranded, or braided. When using flat cable, only the V- is connected to a good earth ground. Use a 1" copper braid or a #8 AWG wire that is less than 3 meters/10 feet in length. This should then be attached to a good earth or building ground (such as an 8 foot stake driven into the ground, attached to building iron or to the cold water plumbing).

If the network is already grounded, do NOT connect the grounding terminal of the tap or ground of the supply to earth. If more than one supply is on the network, then connect the drain wire/shield at ONE supply only, preferably near the physical center of the network.

### **8-3.7 Temporary Terminal Support**

DeviceNet specifies an open-style connector for use on a temporary terminal. Figure 8-3.5 shows a drawing of such a connector.

The connector enables removal and insertion of the terminal cable under power. This allows standardization of the cable necessary for temporary terminal connection.

**Figure 8-3.5 Connector for Temporary Terminal Support**

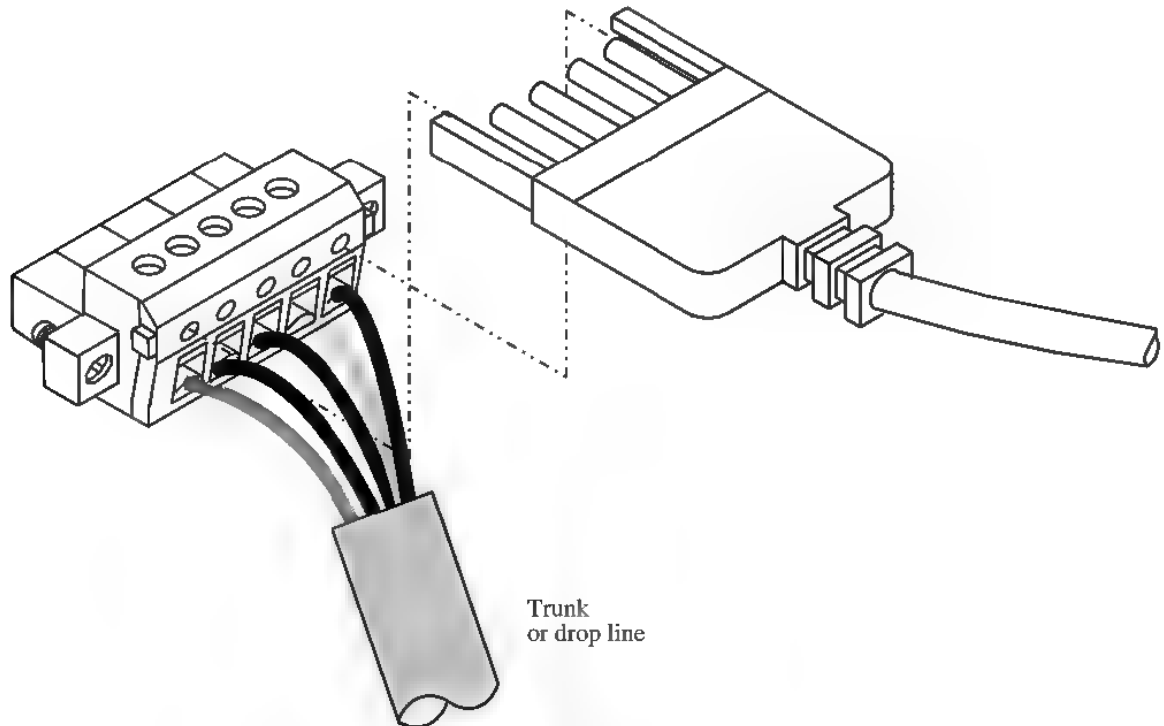
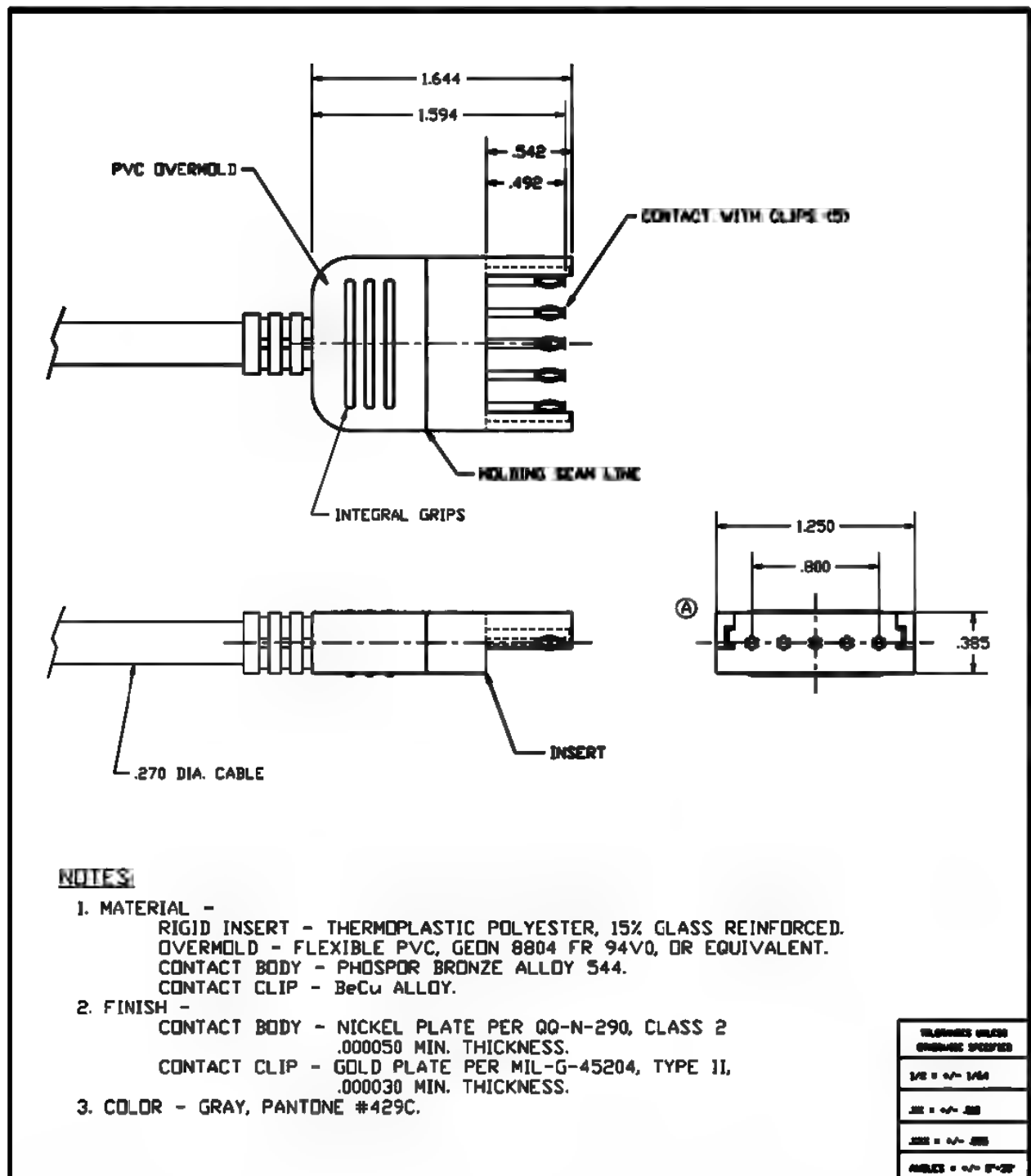




Figure 8-3.6 Connector for Temporary Terminal Support



### 8-3.8 Media

The following cable profiles are more detailed than the specifications found in the Physical Layer Requirements chapter. Included are the following profiles:

- Thick Cable
- Thin Cable
- Flat Cable
- Cable I
- Cable II
- Cable III
- Cable IV
- Cable V
- Cable VI

#### 8-3.8.1 Cable Profile Template

A cable profile defines the Data Pair Specifications, DC Power Pair Specifications, General Specifications, Topology and the physical configuration for the cable. The orientation of the data and power pairs is a requirement of the specification. The following table defines the minimum fields that must be defined for a DeviceNet cable profile.

**Table 8-3.2 Data Pair Specification**

Physical Characteristics	Specification
Conductor pair size	<size> <material>; <#> strands
Insulation diameter	<size>
Colors (CAN_H, CAN_L)	
Pair Twist	<#> / <distance>
Tape shield over pair	<material>
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 500 kHz)
Propagation delay	<#> nSec/ft (maximum)
Capacitance between conductors	<#> pF/ft. at <#> kHz (maximum)
Capacitance between one conductor and other conductor connected to shield.	<#> pF/ft. at <#> kHz (maximum)
Capacitive unbalance	<#> pF/ft at <#> kHz (maximum) ASTM D4566-94
DCR @ 20 °C	<#> Ohms/1000 ft (maximum)
Attenuation:	<#> db/100 ft @ 125 kHz (maximum) <#> db/100 ft @ 250 kHz (maximum) <#> db/100 ft @ 500 kHz (maximum)

**Table 8-3.3 Cable Profile: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	<#> <material>; <#> strands
Insulation diameter	<#>
Colors – (V+, V- )	
Pair twist	<#>/ <distance>
Tape shield over pair	<material>
Electrical Characteristics	Specification
DCR - @ 20 C	<#> Ohms/1000 ft (maximum)

**Table 8-3.4 Cable Profile: General Specification**

Physical Characteristics	Specification
Geometry	
Overall braid shield	<#> % coverage, <#> <material>
Drain wire	<#> <material>; <#> strands
Outside diameter	<size> minimum to <size> maximum
Roundness	Radius delta to be <#> % of O.D.
Jacket marking	Vendor name, part # and additional markings
Electrical Characteristics	Specification
DCR – (braid+tape+drain) 20C	<#> Ohms/1000 ft (maximum)
Applicable Environmental Characteristics	Specification
Agency Certifications	
Flexure	<#> cycles at bend radius, <#> degrees, <#> pull force, <#> cycles/minute, <method>
Bend Radius	<#> x diameter (installation) / <#> x diameter (fixed) <method>
Operating ambient temperature	<#> to <#>C
Storage temperature	<#> to <#> C
Pull tension	<#> lbs
Connector Compatibility	<Open, Mini, Micro....., ....>
Topology Compatibility	<Trunk, Drop, ...>
Unique Characteristics	Application Specific

**Table 8-3.5 Cable Profile: Topology**

Data Rate	Max Cable Distance	Trunk Exchange (Thick Cable)	Cumulative Drop	Maximum Drop
125kb	<#>m (<#>ft)	<#>	<#>m (<#>ft)	<#>m (<#>ft)
250kb	<#>m (<#>ft)	<#>	<#>m (<#>ft)	<#>m (<#>ft)
500kb	<#>m (<#>ft)	<#>	<#>m (<#>ft)	<#>m (<#>ft)

Important: Minimum and maximum lengths may be effected by connector DCR, therefore when defining maximum or minimum lengths of a new cable the connector DCR must be considered.

**Figure 8-3.7 Cable Profile: Physical Configuration of Cable**

**Table 8-3.6 Cable Profile: Max. Current Available (amps) Based on Network Length**

**Figure 8-3.8 Cable Profile: Current Available on the DeviceNet Power Bus**

### 8-3.8.2 Thick Cable Profile

Included are the following specifications regarding Thick Cable:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration
- Available Bus Current

**Table 8-3.7 Thick Cable: Data Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#18 Copper (minimum); 19 strands min (individually tinned)
Insulation diameter	0.150 inches (nominal)
Colors	Light Blue White
Pair Twist/ft	3 (approx.)
Tape shield over pair	2 mil / 1 mil, Al / Mylar Al side out w/shorting fold (pull on applied)
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 1 MHz)
Propagation delay	1.36 nSec/ft (maximum)
Capacitance between conductors	12 pF / ft. at 1 kHz (nominal)
Capacitance between one conductor and other conductor connected to shield.	24 pF / ft. at 1 kHz (nominal)
Capacitive unbalance	1200 pF/1000 ft at 1 kHz (nominal)
DCR - @ 20 deg C	6.9 Ohms/1000 ft (maximum)
Attenuation:	0.13 db/100 ft @ 125 kHz (maximum) 0.25 db/100 ft @ 500 kHz (maximum) 0.40 dB/100ft@1.00MHz (maximum)

**Table 8-3.8 Thick Cable: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#15 Copper (minimum); 19 strands minimum (individually tinned)
Insulation diameter	0.098 inches (nominal)
Colors	Red Black
Pair Twist/ft	3 approximately
Tape shield over pair	1.0 mil/ 1 mil, Al/Mylar Al side out w/shorting fold (pull-on applied)
Electrical Characteristics	Specification
DCR - @ 20 deg C	3.6 Ohms/1000 ft (maximum)

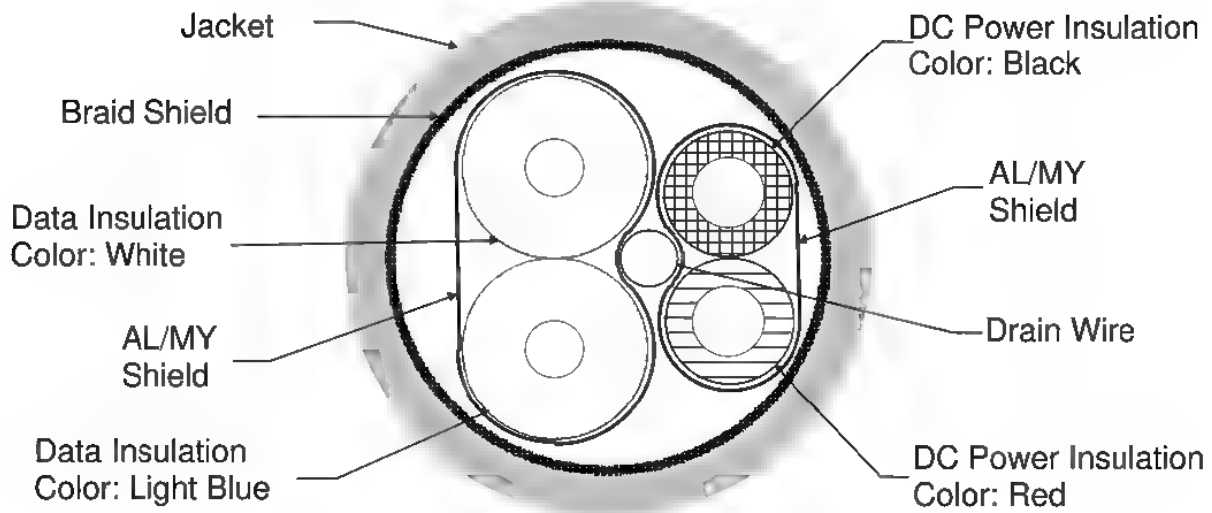
**Table 8-3.9 Thick Cable: General Specification**

<b>Physical Characteristics</b>	<b>Specification</b>
Geometry	Two shielded pairs, Common axis with drain wire in center
Overall braid shield	65% coverage 36 AWG or 0.12mm tinned Cu braid minimum (individually tinned)
Drain wire	#18 Copper min.; 19 Strands min. (individually tinned)
Outside diameter	0.410 inches (min) to 0.490 inches (max )
Roundness	Radius delta to be within 15% of 0.5 O.D.
Jacket marking	Vendor Name & Part #, and additional markings
<b>Electrical Characteristics</b>	<b>Specification</b>
DCR (braid+tape+drain)	1.75 Ohms/1000 ft (nom. @ 20 C)
<b>Applicable Environmental Characteristics</b>	<b>Specification</b>
Agency Certifications (U.S. and Canada)	NEC (UL) type, CL2/CL3 (min.)
Flexure	2000 cycles at bend radius, 90 degrees, 2 lb. Pull force, 15 cycles per minute, Tic Toc or C track method
Bend Radius	20 x diameter(installation) / 7 x diameter(fixed)
Operating ambient temperature	-20 to +60 C @ 8 amps; de-rate current linearly to zero @ 80 C
Storage temperature	-40 to +85 C
Pull tension	190 lbs max.
Connector Compatibility	Mini, Open
Topology Compatibility	Trunk, Drop

**Table 8-3.10 Thick Cable: Topology**

Data Rate	Max Cable Distance	Trunk Exchange (Thick Cable)	Cumulative Drop	Maximum Drop
125kb	500m (1640ft)	1.0	156m (512ft)	6m (20ft)
250kb	250m (820ft)	1.0	78m (256ft)	6m (20ft)
500kb	100m (328ft)	1.0	39m (128ft)	6m (20ft)

**Figure 8-3.9 Thick Cable: Physical Configuration**



**Table 8-3.11 Thick Cable: Max. Current Available (amps) Based on Network Length**

Network Length in meters (feet)	0	25 (82)	50 (164)	100 (328)	150 (492)	200 (656)	250 (820)	300 (984)	350 (1148)	400 (1312)	450 (1476)	500 (1640)
Maximum Current in amps	8.00	8.00	5.42	2.93	2.01	1.53	1.23	1.03	0.89	0.78	0.69	0.63

**Figure 8-3.10 Thick Cable: Current Available on the DeviceNet Power Bus**

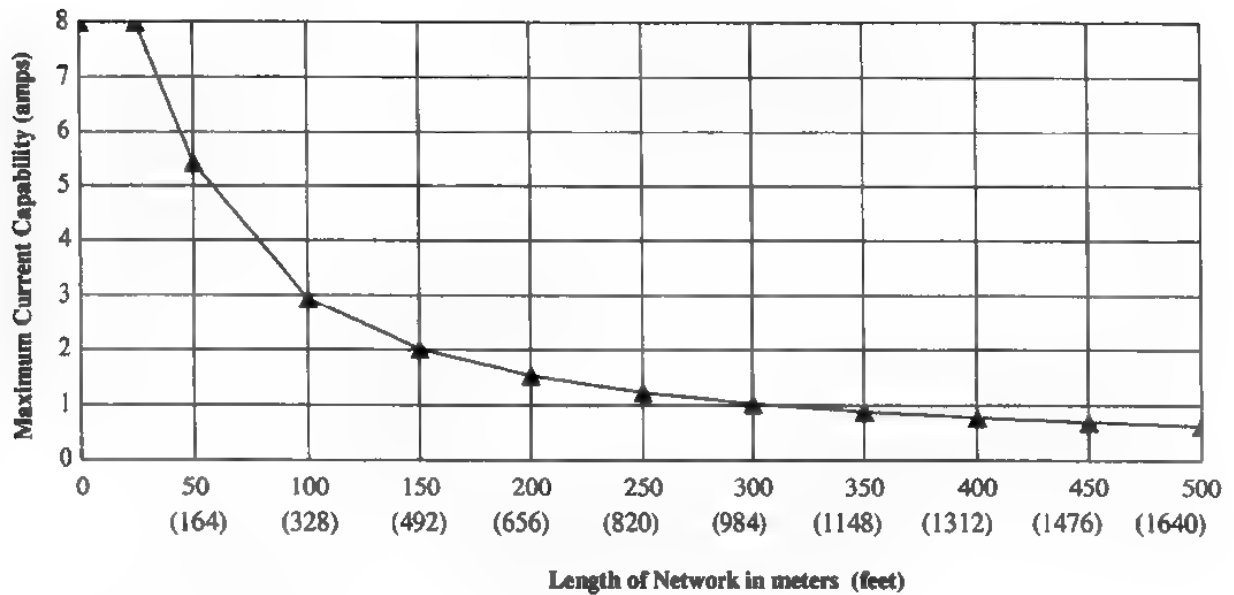


Table 8-3.11 and Figure 8-3.10 are computed by using the formula:

$$I = 4.65V / ((\text{Cable DCR} * \text{Length of Network}) + (\text{Contact DCR} * \text{Number of Contacts})).$$

Where Cable DCR = 0.00445 ohms/ft, Contact DCR = 0.001 ohms, and Number of Contacts = 128 (since each tap has two contacts in series). The Cable DCR is determined using an ambient of 80C, and temperature coefficient of 0.00393 per degree C.



### 8-3.8.3 Thin Cable Profile

Included are the following specifications regarding Thin Cable:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration
- Available Bus Current

**Table 8-3.12 Thin Cable: Data Pair Specifications**

Physical Characteristics	Specification
Conductor pair size	#24 Copper (minimum); 19 strands minimum (individually tinned)
Insulation diameter	0.077 inches (nominal)
Colors	Light Blue White
Pair Twist/ft	5 (approximately)
Tape shield over pair	1 mil / 1 mil, Al / Mylar Al side out w/shorting fold (pull-on applied)
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 1 MHz)
Propagation delay	1.36 nSec/ft (maximum)
Capacitance between conductors	12 pF / ft. at 1 kHz (nominal)
Capacitance between one conductor and other conductor connected to shield	24 pF / ft. at 1 kHz (nominal)
Capacitive unbalance	1,200 pF/1000 ft at 1 kHz (maximum)
DCR - @ 20 C	28 Ohms/1000 ft (maximum)
Attenuation:	0.29 dB/100 ft @ 125 kHz (maximum) 0.50 dB/100 ft @ 500 kHz (maximum) 0.70 dB/100 ft @ 1.00 MHz (maximum)

**Table 8-3.13 Thin Cable: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#22 Copper (minimum), 19 strands minimum (individually tinned)
Insulation diameter	0.055 inches (nominal)
Colors	Red Black
Pair twist/ft	5 approximately
Tape shield over pair	1.0 mil/ 1.0 mil, Al/Mylar Al side out w/shorting fold (pull-on applied)
Electrical Characteristics	Specification
DCR - @ 20 C	17.5 Ohms/1000 ft (maximum)

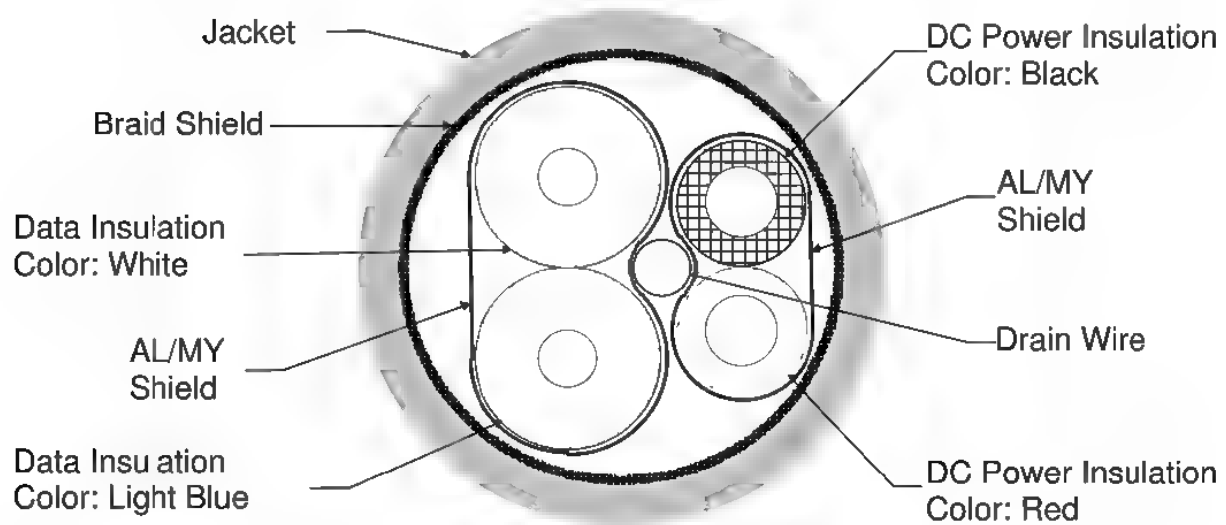
**Table 8-3.14 Thin Cable: General Specification**

<b>Physical Characteristics</b>	<b>Specification</b>
Geometry	Two shielded pairs, Common axis with drain wire in center
Overall braid shield	65% coverage 36 AWG or 0.12mm tinned Cu braid minimum (individually tinned)
Drain wire	#22 Copper 19 strands minimum (individually tinned)
Outside diameter	0.240 inches (min.) to 0.280 inches (max.)
Roundness	Radius delta to be within 20% of 0.5 O.D
Jacket marking	Vendor name and part # and additional markings
<b>Electrical Characteristics</b>	<b>Specification</b>
DCR (braid+tape+drain)	3 2 Ohms/1000 ft (nom. @ 20 C)
<b>Applicable Environmental Characteristics</b>	<b>Specification</b>
Agency Certifications (U.S. and Canada)	NEC (UL) type CL2 (min.)
Flexure	2000 cycles at bend radius, 90 degrees, 2 lb. Pull force, 15 cycles per minute, Tie Toc or C track method
Bend Radius	20 x diameter (installation) / 7 x diameter (fixed)
Operating ambient temperature	-20 to +70 C @ 1.5 amps, de rate current linearly to zero at 80 C
Storage temperature	-40 to +85 C
Pull tension	65 lbs max.
Connector Compatibility	Mini, Micro, Open
Topology Compatibility	Trunk, Drop

**Table 8-3.15 Thin Cable: Topology**

Data Rate	Max Cable Distance	Trunk Exchange (Thick Cable)	Cumulative Drop	Maximum Drop
125kb	100m (328ft)	5.0	156m (512ft)	6m (20ft)
250kb	100m (328ft)	2.5	78m (256ft)	6m (20ft)
500kb	100m (328ft)	1.0	39m (128ft)	6m (20ft)

**Figure 8-3.11 Thin Cable: Physical Configuration**



**Table 8-3.16 Thin Cable: Max. Current Available (amps) Based on Network Length**

Network Length in meters (feet)	0	10 (33)	20 (66)	30 (98)	40 (131)	50 (164)	60 (197)	70 (230)	80 (262)	90 (295)	100 (328)
Maximum Current in amps	3.00	3.00	3.00	2.06	1.57	1.26	1.06	0.91	0.80	0.71	0.64

**Figure 8-3.12 Thin Cable: Current Available on the DeviceNet Power Bus**

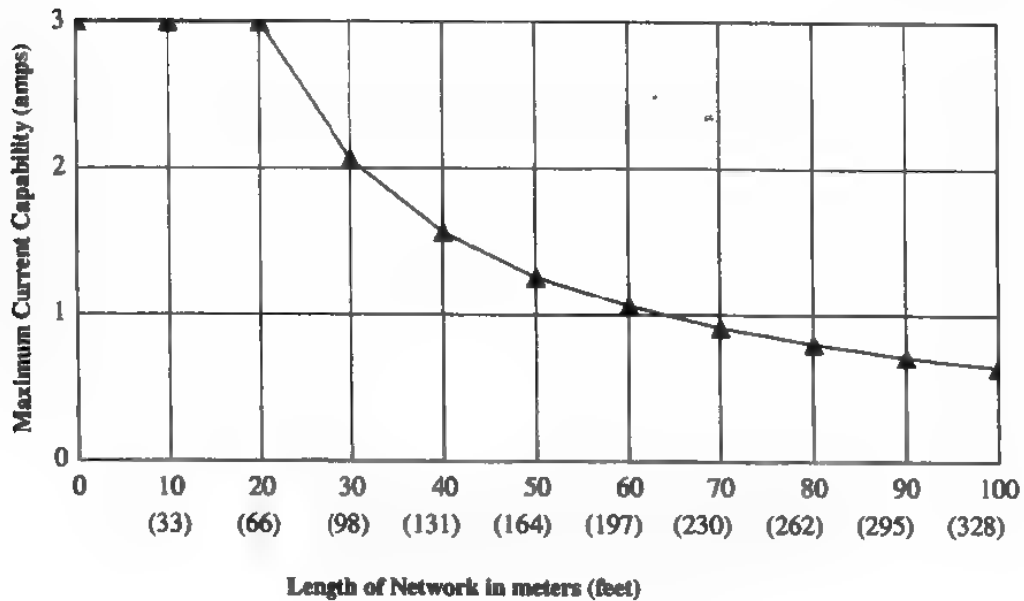


Table 8-3.16 and Figure 8-3.12 are computed by using the formula:

$$I = 4.65V / ((\text{Cable DCR} * \text{Length of Network}) + (\text{Contact DCR} * \text{Number of Contacts})).$$

Where Cable DCR = 0.0216 ohms/ft, Contact DCR = 0.001 ohms, and Number of Contacts = 128 (since each taps has two contacts in series). The Cable DCR is determined using an ambient of 80C, and temperature coefficient of 0.00393 per degree C.

#### 8-3.8.4 Flat Cable Profile

Included are the following specifications regarding Flat Cable:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration
- Available Bus Current

**Table 8-3.17 Flat Cable: Data Pair Specifications**

Physical Characteristics	Specification
Conductor pair size	#16 Copper (minimum); 19 strands minimum (individually tinned)
Insulation diameter	0.110 inches (nominal)
Colors	Light Blue White
Pair Twist/ft	N/A
Tape shield over pair	N/A
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 500 kHz)
Propagation delay	1.60 nSec/ft (maximum)
Capacitance between conductors	14.7 pF/ft at 500 kHz (maximum)
Capacitance between one conductor and other conductor connected to shield.	N/A
Capacitive unbalance	1.2 pF/ft at 500 kHz (maximum) ASTM D4566-94
DCR - @ 20 °C	4.9 Ohms/1000 ft (maximum)
Attenuation:	0.13 dB/100 ft @ 125 kHz (maximum) 0.25 dB/100 ft @ 250 kHz (maximum) 0.40 dB/100 ft @ 500 kHz (maximum)

**Table 8-3.18 Flat Cable: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#16 Copper (minimum); 19 strands minimum (individually tinned)
Insulation diameter	0.110 inches (nominal)
Colors	Red Black
Pair twist/ft	N/A
Tape shield over pair	N/A
Electrical Characteristics	Specification
DCR - @ 20 °C	4.9 Ohms/1000 ft (maximum)

**Table 8-3.19 Flat Cable: General Specification**

<b>Physical Characteristics</b>	<b>Specification</b>
Geometry	N/A
Overall braid shield	N/A
Drain wire	N/A
Outside diameter	See Figure 8-3.13
Roundness	N/A
Jacket marking	Vendor name and part # and additional markings
<b>Electrical Characteristics</b>	<b>Specification</b>
DCR (braid+tape+drain)	N/A
<b>Applicable Environmental Characteristics</b>	<b>Specification</b>
Agency Certifications (U.S. and Canada)	NEC (UL) type CL2 (min.)
Flexure	1.0M cycles at bend radius, 6 feet min length, 15 cycles per minute, C track method
Bend Radius	10 x thickness (installation and fixed)
Operating ambient temperature	-25 to +75 C @ 8 amps; de-rate current linearly to zero at 80 C
Storage temperature	-40 to +85 C
Pull tension	90 lbs max.
Durometer	95 Shore A (maximum)
Connector Compatibility	Flat
Topology Compatibility	Trunk

Table 8-3.20 Flat Cable: Topology

Data Rate	Max Cable Distance	Trunk Exchange (Thick Cable)	Cumulative Drop	Maximum Drop
125kb	420m (1378ft)	n a.	156m (512ft)	6m (20ft)
250kb	200m (656ft)	n a.	78m (256ft)	6m (20ft)
500kb	75m (246ft)	n.a.	39m (128ft)	6m (20ft)

Figure 8-3.13 Flat Cable: Physical Configuration

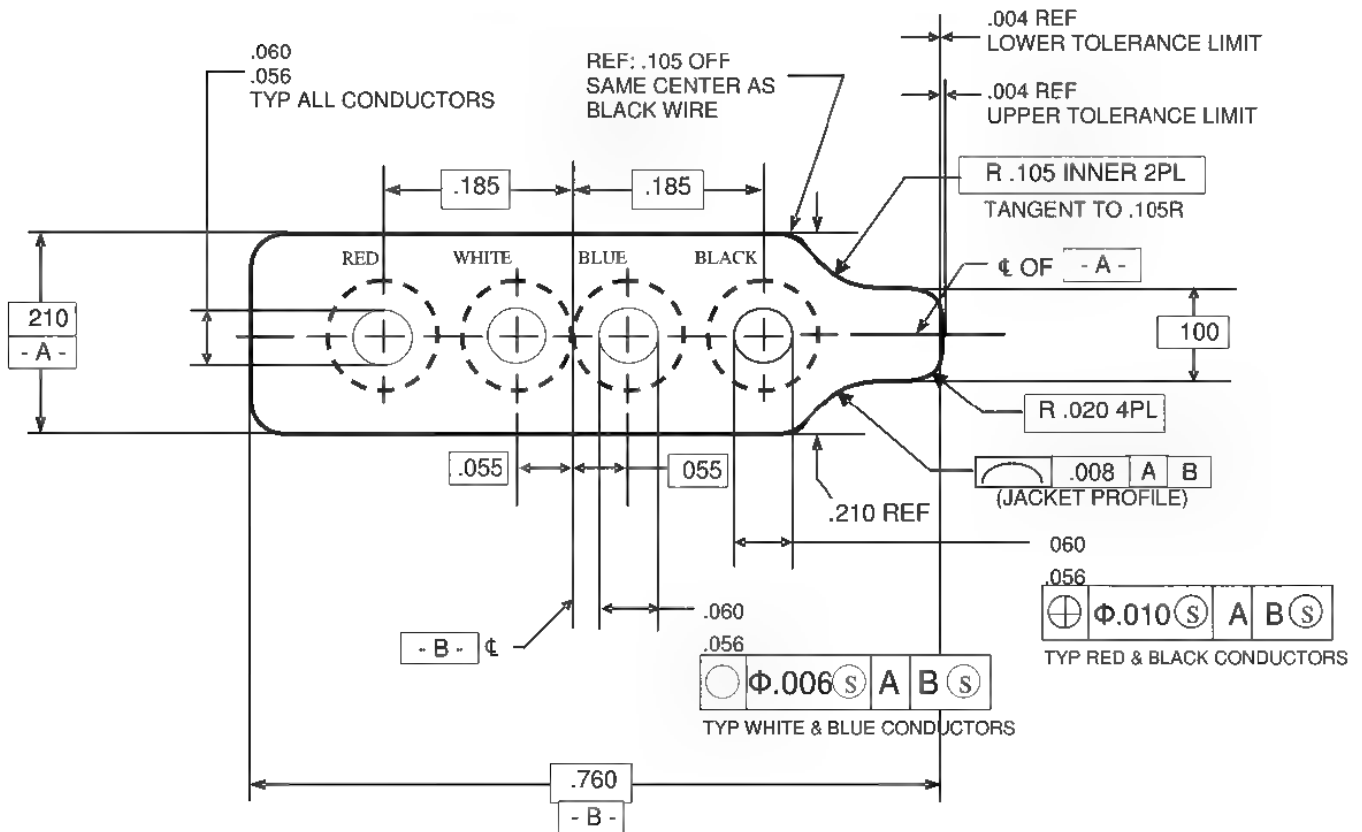


Table 8-3.21 Flat Cable: Max. Current Available (amps) Based on Network Length

Network Length in meters (feet)	0	12.5 (41)	25 (82)	50 (164)	100 (328)	150 (492)	200 (656)	250 (820)	300 (984)	350 (1148)	400 (1312)	420 (1378)
Maximum Current in amps	8.00	8.00	8.00	5.65	2.86	1.91	1.44	1.15	.96	.82	.72	.69

Figure 8-3.14 Flat Cable: Current Available on the DeviceNet Power Bus

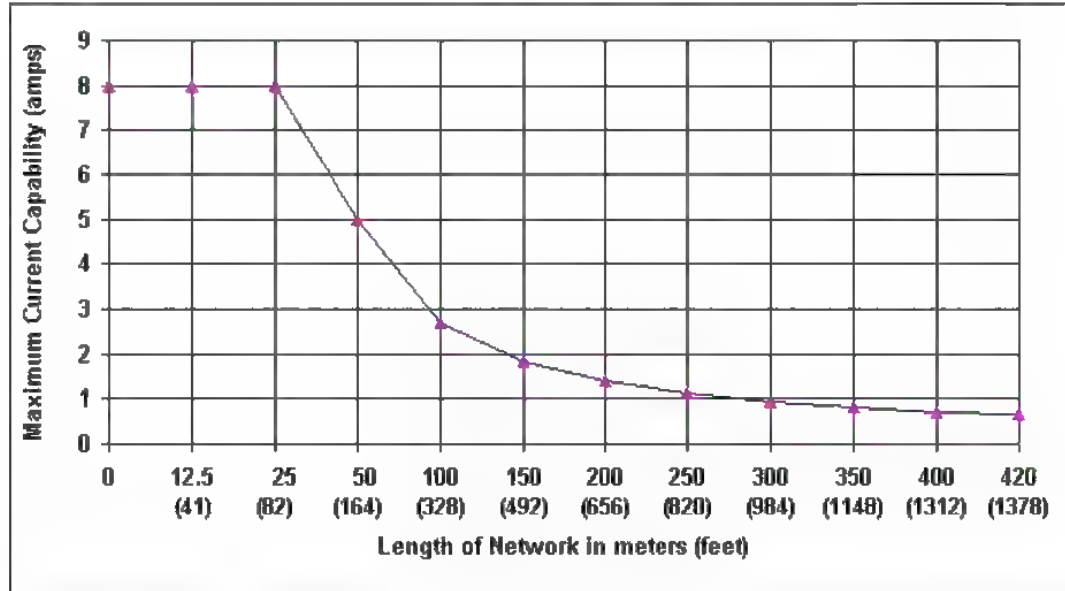


Table 8-3.21 and Figure 8-3.14 are computed by using the formula:

$$I = 4.65V / ((\text{Cable DCR} * \text{Length of Network}) + (\text{Contact DCR} * \text{Number of Contacts})).$$

Where Cable DCR = 0.0049 ohms/ft, Contact DCR = 0.010 ohms, and Number of Contacts = 2 (since Flat Media taps installation does not put Contact DCR in series). The Cable DCR is as specified at an ambient of 20C.



### 8-3.8.5 Flat II Cable Profile

Included are the following specifications regarding Flat II Cable:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration
- Available Bus Current

**Table 8-3.22 Flat II Cable: Data Pair Specification**

Physical Characteristics	Specification
Conductor pair size	0.5mm <sup>2</sup> max or 20AWG max Cu (Sn plating), 20 max Strands of 0.18mm min, 1 twist / 15mm
Insulation diameter	2.54 mm ±0.06 mm
Colors	White Blue
Pair Twist	None
Tape shield over pair	None
Electrical Characteristics	Specification
Impedance	120 Ohms +/-10% (at 500 kHz)
Propagation delay	Maximum 1.78 nSec/ft (maximum)
Capacitance between conductors	15.8 pF/ft +5% max (at 1kHz, 20degC)
Capacitance between one conductor and other conductor connected to shield	None
Capacitive unbalance	2.07pF/ft ±5% ASTM D4566
DCR - @ 20 deg C	10.6 Ohms/1000ft (MAX) UL1581
Attenuation	dB/100ft (at 20 deg C) 0.14dB max. @ 125 kHz 0.25dB max. @ 250 kHz 0.42dB max. @ 500 kHz 0.72dB max. @ 1.0 MHz

**Table 8-3.23 Flat II Cable: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	0.75mm <sup>2</sup> max or 18AWG max; Cu (Sn plating) 30 Strands max. of 0.18mm min., twist 1twist / 20mm
Insulation diameter	2.54mm±0.06mm
Colors	Red Black
Pair Twist	None
Tape shield over pair	None
Electrical Characteristics	Specification
DCR - @ 20 deg C	6.9 Ohms/1000ft (MAX.)

**Table 8-3.24 Flat II Cable: General Specification**

Physical Characteristics	Specification
Geometry	Flat
Over braid shield	None
Drain wire	None
Outside diameter	Width: 10.16mm +0mm,-0.5mm; Height: 2.54 mm +/-0.6mm
Roundness	Flat
Jacket marking	Manufacturer name, and Certification mark
Electrical Characteristics	Specification
DCR - @ 20 deg C	None
Applicable Environmental Characteristics	Specification
Agency Certification (U S. and Canada)	To meet local regulatory agencies.
Flexure	Specified by vendor //under measurement
Bend Radius	Specified by vendor
Operating ambient temperature	-10 to 55 deg C
Storage temperature	-20 to 65 deg C
Pull tension	40 lbs maximum
Connector Compatibility	FLAT II TRUNK CONNECTOR Female and Male
Topology Compatibility	Trunk, Branch

**Figure 8-3.15 Flat II Cable: Topology**

Data Rate	Max Cable Distance	Trunk Exchange (Thick cable)	Cumulative Drop	Maximum Drop
125kb	265m (1050ft)	n.a.	135 (511ft)	6m (20ft)
250kb	150m (574ft)	n.a.	48 (157ft)	6m (20ft)
500kb	75m (213ft)	n.a.	35 (82ft)	6m (20ft)

**Figure 8-3.16 Flat II Cable Physical Configuration**

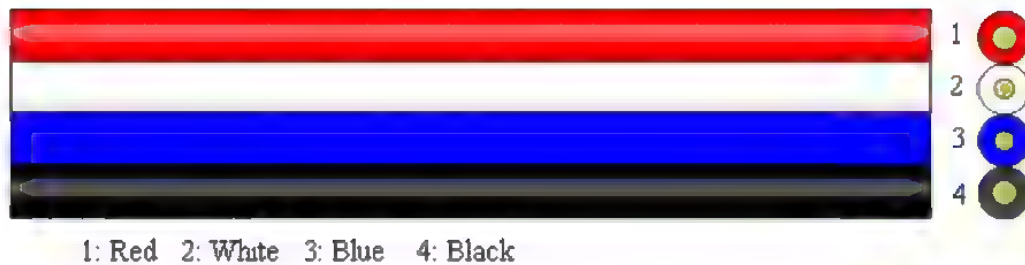


Figure 8-3.17 Dimension of Flat II cable

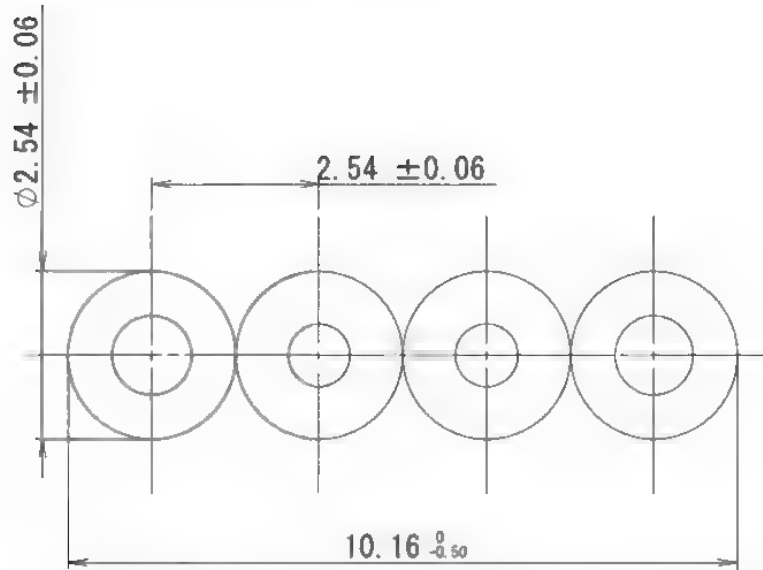
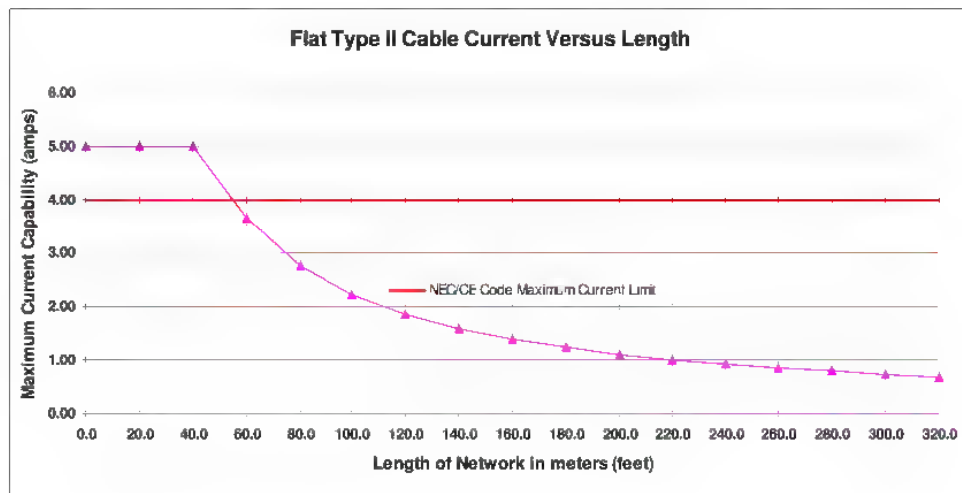


Table 8-3.25 Flat II Cable: Max. Current Available (amps) Based on Network Length

Network Length in meters (feet)	0	20 (66)	40 (131)	80 (262)	100 (328)	140 (459)	200 (656)	260 (853)	300 (984)	320 (1050)
Maximum Current in amps	5	5	5	2.75	2.21	1.58	1.11	.85	.74	.69

Figure 8-3.18 Flat II Cable: Current Available on the DeviceNet Power Bus



### 8-3.8.6 Cable I Profile

Included are the following specifications regarding Cable I:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration
- Available Bus Current

**Table 8-3.26 Cable I: Data Pair Specifications**

Physical Characteristics	Specification
Conductor pair size	#24 Copper (minimum); 19 strands minimum (individually tinned)
Insulation diameter	0.077 inches (nominal)
Colors	Light blue White
Pair Twist/ft	5 (approximately)
Tape shield over pair	1 mil / 1 mil, Al / Mylar Al side out w/shorting fold (pull-on applied)
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 1 MHz)
Propagation delay	1.36 nSec/ft (maximum)
Capacitance between conductors	12 pF / ft. at 1 kHz (nominal)
Capacitance between one conductor and other conductor connected to shield	24 pF / ft. at 1 kHz (nominal)
Capacitive unbalance	1,200 pF/1000 ft at 1 kHz (maximum)
DCR - @ 20 C	28 Ohms/1000 ft (maximum)
Attenuation.	0.29 dB/100 ft @ 125 kHz (maximum) 0.50 dB/100 ft @ 500 kHz (maximum) 0.70 dB/100 ft @ 1.00 MHz (maximum)

**Table 8-3.27 Cable I: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#22 Copper (minimum), 19 strands minimum (individually tinned)
Insulation diameter	0.055 inches (nominal)
Colors	Red Black
Pair twist/ft	5 approximately
Tape shield over pair	1.0 mil/ 1.0 mil, Al/Mylar Al side out w/shorting fold (pull-on applied)
Electrical Characteristics	Specification
DCR - @ 20 C	17.5 Ohms/1000 ft (maximum)

**Table 8-3.28 Cable I: General Specification**

<b>Physical Characteristics</b>	<b>Specification</b>
Two shielded pairs	Common axis with drain wire in center
Overall braid shield	Overall tape shield (1 mil/1 mil, Al/Mylar) or tinned copper stranded shield (65% minimum coverage)
Drain wire	#22 Copper 19 strands minimum (individually tinned)
Outside diameter	Specified by vendor
Roundness	Radius delta to be within 20% of 0.5 O.D
Jacket marking *	Vendor name and part # and additional markings
<b>Electrical Characteristics</b>	<b>Specification</b>
DCR (braid+tape+drain)	3.2 Ohms/1000 ft (nom. @ 20 C)
<b>Applicable Environmental Characteristics</b>	<b>Specification</b>
Agency Certifications (U.S. and Canada)	Compliant with local government regulations.
Flexure	Specified by vendor
Bend Radius	Specified by vendor
Operating ambient temperature	-20 to +70 C @ 1.5 amps; de-rate current linearly to zero at 80 C (minimum).
Storage temperature	-40 to +85 C (minimum).
Pull tension	65 lbs max.

Other types of jacket insulation are allowable provided that internal construction and electrical characteristics adhere to this specification.

**Table 8-3.29 Cable I: Topology**

Data Rate	Max Cable Distance	Trunk Exchange (Thick Cable)	Cumulative Drop	Maximum Drop
125kb	100m (328ft)	5.0	156m (512ft)	6m (20ft)
250kb	100m (328ft)	2.5	78m (256ft)	6m (20ft)
500kb	100m (328ft)	1.0	39m (128ft)	6m (20ft)

**Figure 8-3.19 Cable I: Physical Configuration**

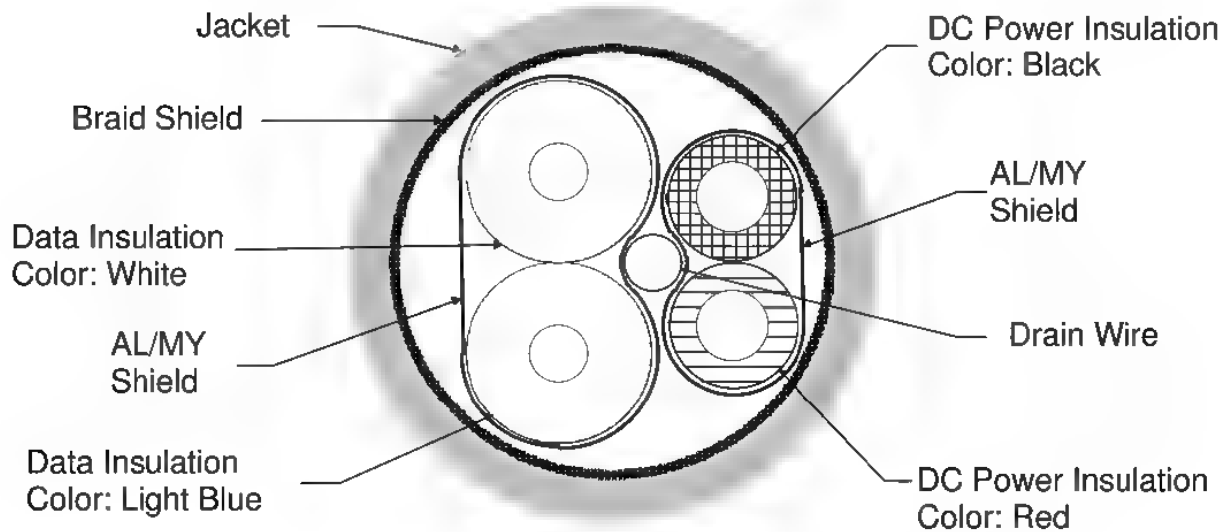


Table 8-3.30 Cable I: Max. Current Available (amps) Based on Network Length

Network Length in meters (feet)	0	10 (33)	20 (66)	30 (98)	40 (131)	50 (164)	60 (197)	70 (230)	80 (262)	90 (295)	100 (328)
Maximum Current in amps	3.00	3.00	3.00	2.06	1.57	1.26	1.06	0.91	0.80	0.71	0.64

Figure 8-3.20 Cable I: Current Available on the DeviceNet Power Bus

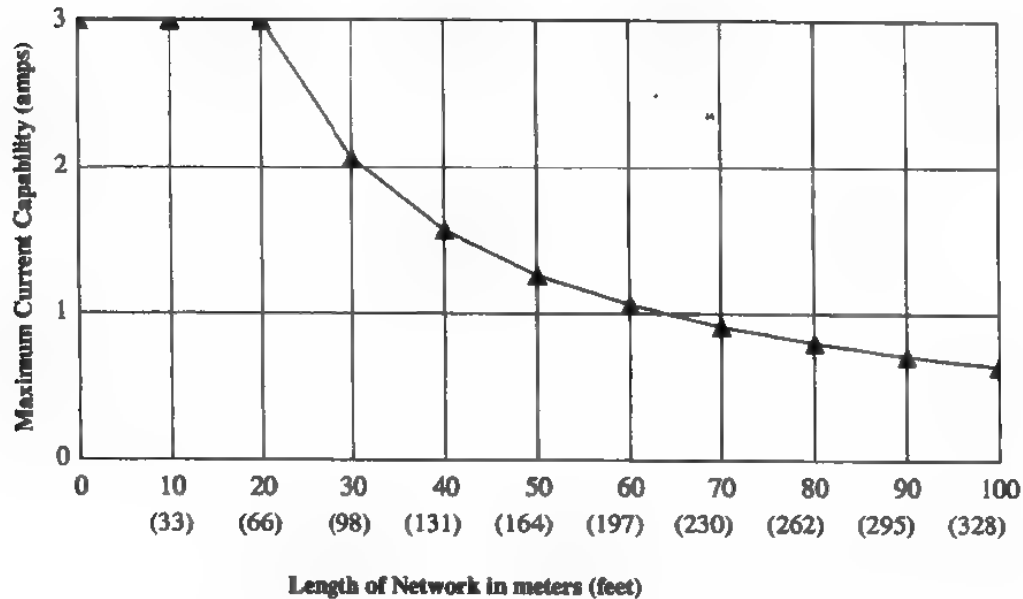


Table 8-3.30 and Figure 8-3.20 are computed by using the formula:

$$I = 4.65V / ((\text{Cable DCR} * \text{Length of Network}) + (\text{Contact DCR} * \text{Number of Contacts})).$$

Where Cable DCR = 0.0216 ohms/ft, Contact DCR = 0.001 ohms, and Number of Contacts = 128 (since each taps has two contacts in series). The Cable DCR is determined using an ambient of 80C, and temperature coefficient of 0.00393 per degree C.

### 8-3.8.7 Cable II Profile

Included are the following specifications regarding Cable II:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration
- Available Bus Current
- 

**Table 8-3.31 Cable II: Data Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#18 Copper (minimum), 19 strands min (individually tinned)
Insulation diameter	0.150 inches (nominal)
Colors	Light blue White
Pair Twist/ft	3 (approx.)
Tape shield over pair	2 mil / 1 mil, Al / Mylar Al side out w/shorting fold (pull-on applied)
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 1 MHz)
Propagation delay	1.36 nSec/ft (maximum)
Capacitance between conductors	12 pF / ft at 1 kHz (nominal)
Capacitance between one conductor and other conductor connected to shield	24 pF / ft. at 1 kHz (nominal)
Capacitive unbalance	1200 pF/1000 ft at 1 kHz (nominal)
DCR - @ 20 deg C	6.9 Ohms/1000 ft (maximum)
Attenuation:	0.13 db/100 ft @ 125 kHz (maximum) 0.25 db/100 ft @ 500 kHz (maximum) 0.40 db/100ft@1.00MHz (maximum)

**Table 8-3.32 Cable II: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#15 Copper (minimum); 19 strands minimum (individually tinned)
Insulation diameter	0.098 inches (nominal)
Colors	Red Black
Pair Twist/ft	3 approximately
Tape shield over pair	1.0 mil/ 1 mil, Al/Mylar Al side out w/shorting fold (pull-on applied)
Electrical Characteristics	Specification
DCR - @ 20 deg C	3.6 Ohms/1000 ft (maximum)



**Table 8-3.33 Cable II: General Specification**

<b>Physical Characteristics</b>	<b>Specification</b>
Two shielded pairs	Common axis with drain wire in center
Overall shield	Overall tape shield (1 mil/1 mil, Al/Mylar) or tinned copper stranded shield (65% minimum coverage)
Drain wire	#18 Copper min.; 19 Strands min (individually tinned)
Outside diameter	Specified by vendor
Roundness	Radius delta to be within 15% of 0.5 O.D
Jacket marking	Vendor Name & Part #, and additional markings
<b>Electrical Characteristics</b>	<b>Specification</b>
DCR (braid+tape+drain)	1.75 Ohms/1000 ft (nom. @ 20 C)
<b>Applicable Environmental Characteristics</b>	<b>Specification</b>
Agency Certifications (U.S and Canada)	Compliant with local government regulations.
Flexure	Specified by vendor
Bend Radius	Specified by vendor
Bend Radius - installation / fixed	Suitable for application.
Operating ambient temperature	-20 to +60 C @ 8 amps; de-rate current linearly to zero @ 80 C (minimum).
Storage temperature	-40 to +85 C (minimum).
Pull tension	190 lbs max

Other types of jacket insulation are allowable provided that internal construction and electrical characteristics adhere to this specification.

**Table 8-3.34 Cable II: Topology**

Data Rate	Max Cable Distance	Trunk Exchange (Thick Cable)	Cumulative Drop	Maximum Drop
125kb	500m (1640ft)	1.0	156m (512ft)	6m (20ft)
250kb	250m (820ft)	1.0	78m (256ft)	6m (20ft)
500kb	100m (328ft)	1.0	39m (128ft)	6m (20ft)

**Figure 8-3.21 Cable II: Physical Configuration**

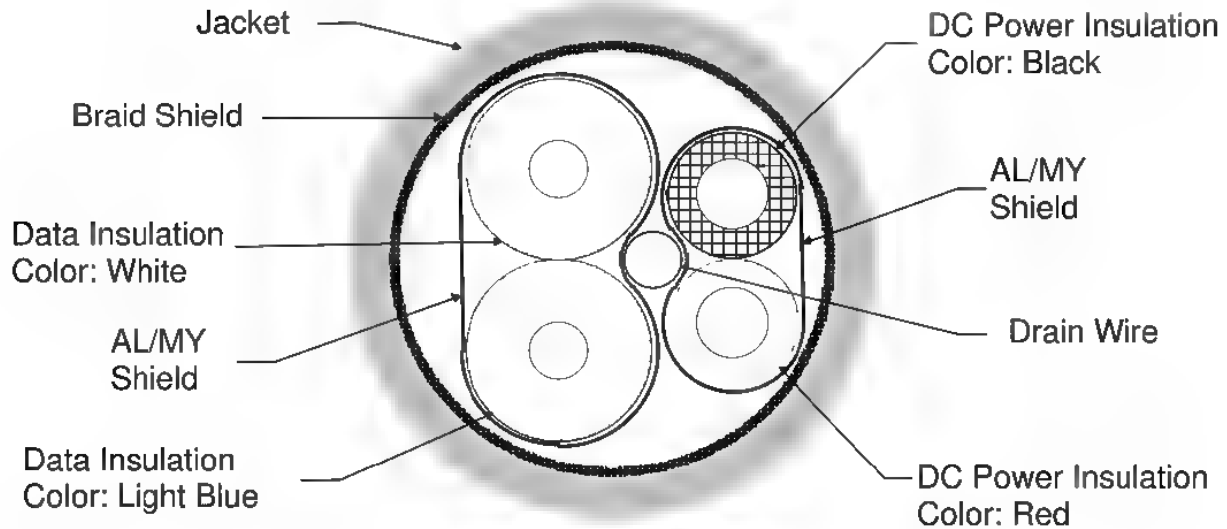


Table 8-3.35 Cable II: Max. Current Available (amps) Based on Network Length

Network Length in meters (feet)	0	25 (82)	50 (164)	100 (328)	150 (492)	200 (656)	250 (820)	300 (984)	350 (1148)	400 (1312)	450 (1476)	500 (1640)
Maximum Current in amps	8.00	8.00	5.42	2.93	2.01	1.53	1.23	1.03	0.89	0.78	0.69	0.63

Figure 8-3.22 Cable II: Current Available on the DeviceNet Power Bus

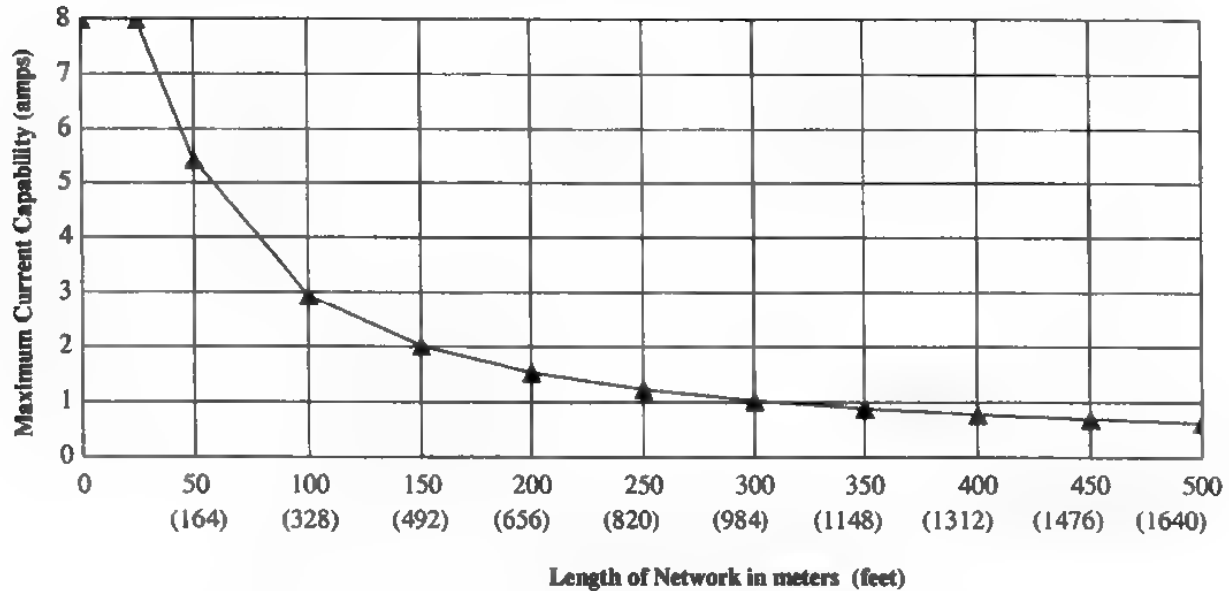


Table 8-3.35 and Figure 8-3.22 are computed by using the formula:

$$I = 4.65V / ((\text{Cable DCR} * \text{Length of Network}) + (\text{Contact DCR} * \text{Number of Contacts})).$$

Where Cable DCR = 0.00445 ohms/ft, Contact DCR = 0.001 ohms, and Number of Contacts = 128 (since each tap has two contacts in series). The Cable DCR is determined using an ambient of 80C, and temperature coefficient of 0.00393 per degree C.

### 8-3.8.8 Cable III Profile

Included are the following specifications regarding Cable III:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration

**Table 8-3.36 Cable III: Data Pair Specifications**

Physical Characteristics	Specification
Conductor pair size	#20 Copper (minimum); 19 strands (minimum)(individually tinned)
Insulation diameter	.098 inches (nominal)
Colors (CAN High, CAN Low)	White, Light blue
Pair Twist	5/foot (approximately)
Tape shield over pair	1 mil / 1 mil, Al / Mylar
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 1 MHz)
Propagation delay	1.36 nSec/ft (maximum)
Capacitance between conductors	12 pF / ft @ 1 KHz (nominal)
Capacitance between one conductor and other conductor connected to shield.	12 pF / ft. @ 1 KHz (nominal)
Capacitive unbalance	2 1200 pF / 1000 ft at 1 kHz (nominal)
DCR - @ 20 °C	3 10.9 Ohms / 1000 ft (maximum)
Attenuation:	0 29 dB/100 ft @ 125 kHz (maximum) 0 50 dB/100 ft @ 500 kHz (maximum) 4 0.70 dB/100 ft @ 1.00MHz (maximum)

**Table 8-3.37 Cable III: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#18 Copper (minimum); 19 strands (minimum)
Insulation diameter	.060 inches (nominal)
Colors (V+, V-)	Red Black
Pair twist	5 5 / ft. (approximately)
Tape shield over pair	6 1 mil / 1 mil, Al / Mylar
Electrical Characteristics	Specification
7 DCR - @ 20 °C	8 6.9 Ohms / 1000 ft (maximum)

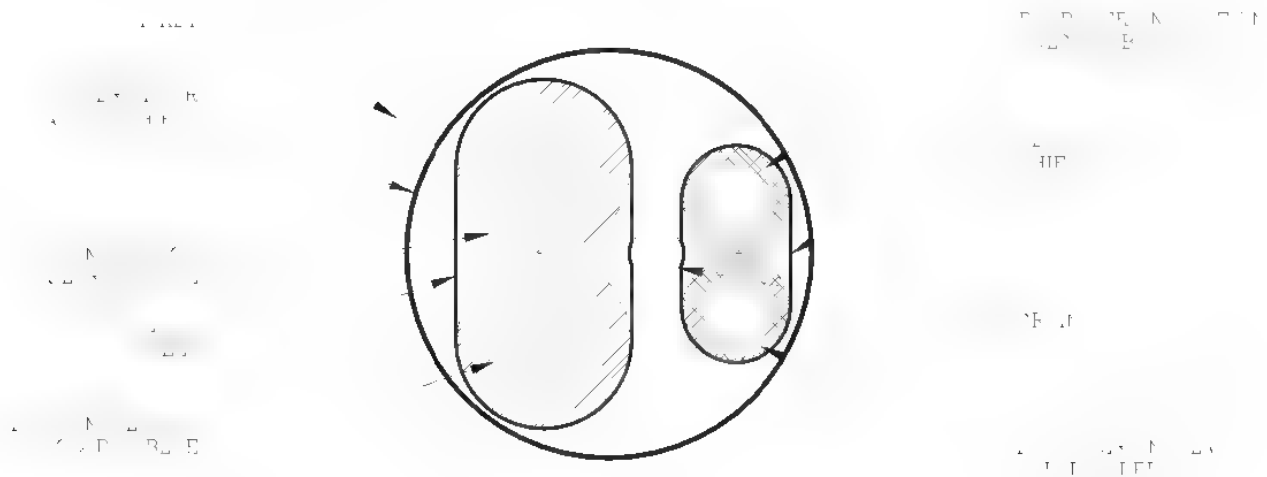
**Table 8-3.38 Cable III: General Specification**

Physical Characteristics	Specification
9 Geometry	10 Common axis with drain wire in center
Overall braid shield	Overall tinned copper stranded shield (65% minimum coverage) 36 AWG minimum
Drain wire size	#20 Copper (minimum); 19 strands (minimum)(individually tinned)
Outside diameter	11 Specified by vendor.
Roundness	12 Radius delta to be within 20% of 0.5 O.D.
Jacket marking/flag label	13 Vendor name, part # and additional markings.
Electrical Characteristics	Specification
14 DCR (braid+tape+drain)	15 3.2 Ohms / 1000 ft (nominal @ 20 °C)
Applicable Environmental Characteristics	Specification
Agency Certifications (U S and Canada)	16 Compliant with local government regulations.
Flexure	17 Suitable for application.
Bend Radius	18 Suitable for application.
Operating temperature	19 -20 to +70 °C @ 1.5 amps; de-rate current 20 linearly to zero @ 80 °C (minimum).
Storage temperature	21 -40 to +85 °C (minimum).
Pull tension	22 65 lbs
Connector Compatibility	23 Mini, micro, open

**Table 8-3.39 Cable III: Topology**

Data Rate	Max Cable Distance	Cumulative Drop	Maximum Drop
125kb	300m	156m (512ft)	6m (20ft)
250kb	250m	78m (256ft)	6m (20ft)
500kb	100m	39m (128ft)	6m (20ft)

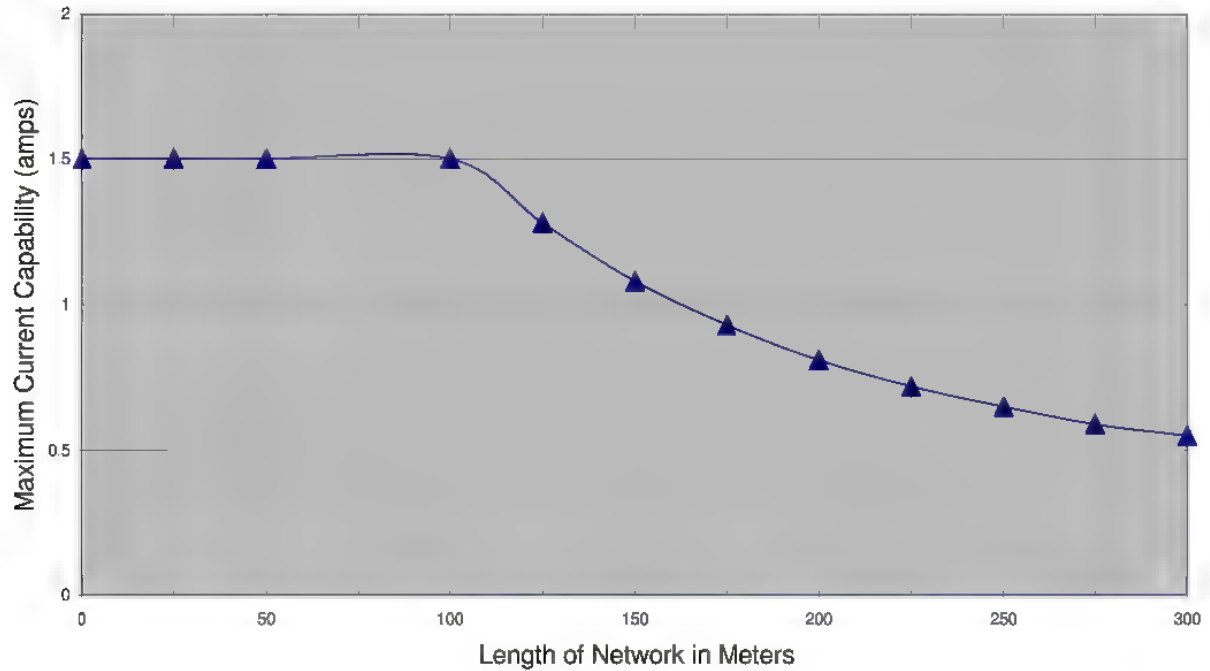
**Figure 8-3.23 Cable III: Physical Configuration**



**Table 8-3.40 Cable III: Max. Current Available (amps) Based on Network Length**

Network Length In Meters [feet]	0	25	50	100	125	150	175	200	225	250	275	300
		[82]	[164]	[328]	[410]	[492]	[574]	[656]	[738]	[820]	[902]	[984]
24 Maximum	1.5	1.5	1.5	1.5	1.28	1.08	0.93	0.81	0.72	0.65	0.59	0.55
25 Current in amps												

**Figure 8-3.24 Cable III: Current Available on the DeviceNet Power Bus**



### 8-3.8.9 Cable IV Profile

Included are the following specifications regarding Cable IV:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration

**Table 8-3.41 Cable IV: Data Pair Specifications**

Physical Characteristics	Specification
Conductor pair size	18 Gauge 19 strand minimum (must be individually tinned)
Insulation diameter	To be specified by vendor
Colors (CAN High, CAN Low)	White, Light blue
Pair Twist	2.6 Twists/foot minimum
Tape shield over pair	None
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 500 kHz)
Propagation delay	1.6 nSec/ft (maximum)
Capacitance between conductors	14.7 pF / ft. @ 500 kHz maximum
Capacitance between one conductor and other conductor connected to shield.	n/a
Capacitive unbalance	1.2 pF/ft at 500 kHz (maximum) ASTM D4566-94
DCR @ 20 °C	6.9 Ohms/1000 ft maximum
Attenuation:	0.13 dB/100 ft @ 125 kHz (maximum) 0.25 dB/100 ft @ 500 kHz (maximum) 0.40 dB/100 ft @ 1.00 MHz (maximum)

**Table 8-3.42 Cable IV: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	16 AWG 19 strand (must be individually tinned)
Insulation diameter	To be specified by vendor
Colors (V+, V-)	Red Black
Pair twist	2.6 twists/ft minimum
Tape shield over pair	n/a
Electrical Characteristics	Specification
DCR @ 20 °C	4.5 Ohms/1000 ft (maximum)



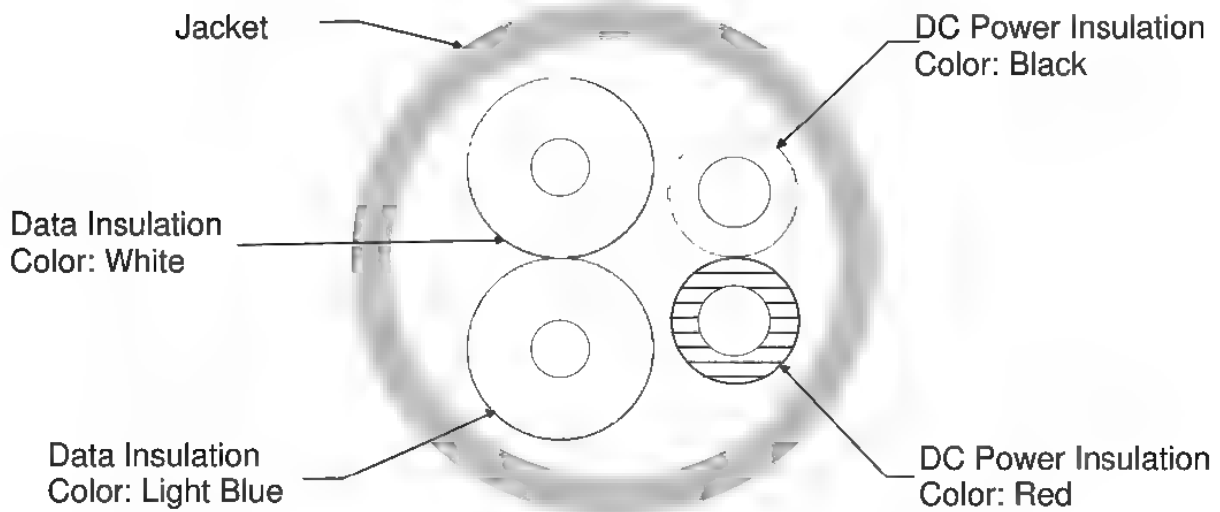
**Table 8-3.43 Cable IV: General Specification**

<b>Physical Characteristics</b>	<b>Specification</b>
Geometry	n/a
Overall braid shield	n/a
Drain wire	n/a
Outside diameter	.44 inches (max)
Roundness	20% maximum of OD
Jacket marking/flag label	Vendor name and part # and additional markings as required by certifying agencies
<b>Electrical Characteristics</b>	<b>Specification</b>
DCR (braid+tape+drain)	n/a
<b>Applicable Environmental Characteristics</b>	<b>Specification</b>
Agency Certifications (U.S. and Canada)	NEC UL type CL2 minimum, To meet local regulatory agencies
Flexure	To be specified by vendor
Bend Radius	To be specified by vendor
Operating temperature	0 to +70 degrees C minimum
Storage temperature	-40 to +85 degrees C minimum
Pull tension	40 lbs. (maximum)
Connector Compatibility	To be specified by vendor

Table 8-3.44 Cable IV: Topology

Data Rate	Max Cable Distance	Cumulative Drop	Maximum Drop
125kb	n/a	156m (512ft)	6m (20ft)
250kb	n/a	78m (256ft)	6m (20ft)
500kb	n/a	39m (128ft)	6m (20ft)

Figure 8-3.25 Cable IV: Physical Configuration



### 8-3.8.10 Cable V Profile

Included are the following specifications regarding Cable V:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration
- Available Bus Current

**Table 8-3.45 Cable V: Data Pair Specifications**

Physical Characteristics	Specification
Conductor pair size	18 AWG 7 strand minimum (must be individually tinned)
Insulation diameter	To be specified by vendor
Colors (CAN High, CAN Low)	White, Light Blue
Pair Twist	2 Twists/foot minimum
Tape shield over pair	2.0 mil/1.0 mil Alum/Mylar, Alum side out with shorting fold
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 500 kHz)
Propagation delay	1.6 nSec/ft (maximum)
Capacitance between conductors	14.7 pF / ft. @ 500 khz (maximum)
Capacitance between one conductor and other conductor connected to shield	29.4 pf/ft @ 500 Khz (maximum)
Capacitive unbalance	1200 pF/1000 ft at 500 kHz (nominal)
DCR - @ 20 C	6.9 Ohms/1000 ft (maximum)
Attenuation:	0.13 db/100 ft @ 125 kHz (maximum) 0.25 db/100 ft @ 500 kHz (maximum) 0.40 dB/100 ft @ 1.00 MHz (maximum)

**Table 8-3.46 Cable V: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	16 AWG 7 strand (must be individually tinned)
Insulation diameter	To be specified by vendor
Colors (V+, V-)	Red Black
Pair twist	2 twists/ft minimum
Tape shield over pair	1.0 mil / 1.0 mil Alum/Mylar, Alum side out with shorting fold
Electrical Characteristics	Specification
DCR - @ 20 C	4.5 Ohms/1000 ft (maximum)

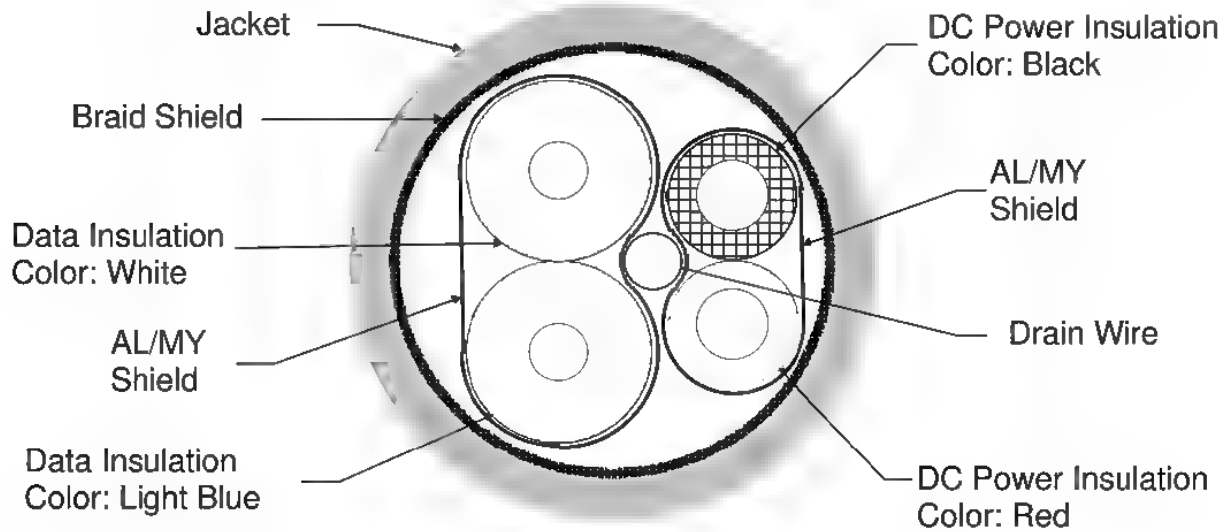
**Table 8-3.47 Cable V: General Specification**

<b>Physical Characteristics</b>	<b>Specification</b>
Geometry	Two shielded pairs, common axis with drain wire in center
Overall braid shield	65% coverage, 36 AWG tinned copper braid (minimum) (individually tinned)
Drain wire	18 AWG 7-strand (minimum) (individually tinned)
Outside diameter	.410 inches (minimum) to .535 inches (maximum)
Roundness	20% maximum of O. D
Jacket marking/flag label	Vendor name and part # and additional markings as required by certifying agencies
<b>Electrical Characteristics</b>	<b>Specification</b>
DCR (braid+tape+drain)	1.75 Ohms/ 1000 ft. (maximum @ 20C)
<b>Applicable Environmental Characteristics</b>	<b>Specification</b>
Agency Certifications (U.S. and Canada)	NEC UL type CL1 (minimum) or applicable local regulatory agencies
Flexure	To be specified by vendor
Bend Radius	To be specified by vendor
Operating temperature	-20 to +75 degrees C
Storage temperature	-40 to +85 degrees C minimum
Pull tension	190 lbs. (maximum)
Connector Compatibility	Mini, Open
Topology Compatibility	Trunk, Drop

**Table 8-3.48 Cable V: Topology**

Data Rate	Max Cable Distance	Trunk Exchange (Thick Cable)	Cumulative Drop	Maximum Drop
125kb	420m (1378ft)	1.0	156m (512ft)	6m (20ft)
250kb	200m (656ft)	1.0	78m (256ft)	6m (20ft)
500kb	75m (246ft)	1.0	39m (128ft)	6m (20ft)

**Figure 8-3.26 Cable V: Physical Configuration**



**Table 8-3.49 Cable V: Max. Current Available (amps) Based on Network Length**

Network Length in meters (feet)	0	12.5 (41)	25 (82)	50 (164)	100 (328)	150 (492)	200 (656)	250 (820)	300 (984)	350 (1148)	400 (1312)	420 (1378)
Maximum Current in amps	8.00	8.00	8.00	5.65	2.86	1.91	1.44	1.15	.96	.82	.72	.69

**Figure 8-3.27 Cable V: Current Available on the DeviceNet Power Bus**

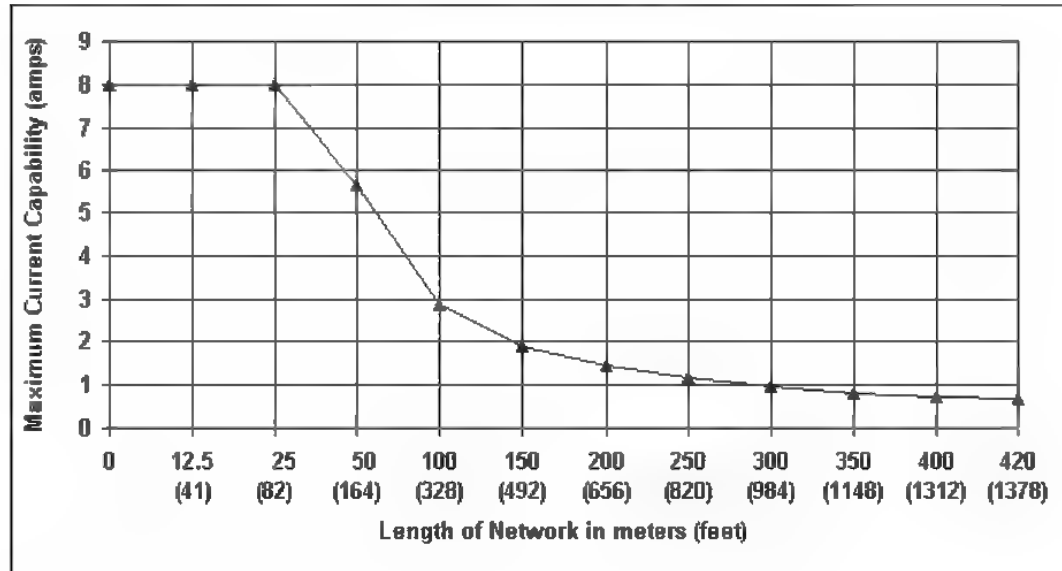


Table 8-3.49 and Figure 8-3.27 are computed by using the formula:

$$I = 4.65V / ((\text{Cable DCR} * \text{Length of Network}) + (\text{Contact DCR} * \text{Number of Contacts})).$$

Where Cable DCR = 0.0049 ohms/ft, Contact DCR = 0.001 ohms, and Number of Contacts = 128 (since Flat Media taps installation does not put Contact DCR in series). The Cable DCR is as specified at an ambient of 20C.

### 8-3.8.11 Cable VI Profile

Included are the following specifications regarding Cable VI:

- Data Pair Specifications
- Power Pair Specifications
- General Specifications
- Topology
- Physical Configuration

**Table 8-3.50 Cable VI: Data Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#26 AWG Copper (minimum); 19 strands minimum (individually tinned)
Insulation diameter	.055 inches (minimum)
Colors	White, Light Blue
Pair Twist	5/foot (approximately)
Tape shield over pair	1 mil/ 1 mil, AL/Mylar
Electrical Characteristics	Specification
Impedance	120 Ohms +/- 10% (at 1 MHz)
Propagation delay	1.36 nSec/ft (maximum)
Capacitance between conductors	12 pF/ft +/- 10% at 1kHz
Capacitance between one conductor and other conductor connected to shield	24 pF/ft at 1kHz (maximum)
Capacitive unbalance	1200 pF/1000 ft at 1kHz (maximum)
DCR - @ 20 deg C	44.4 Ohms/1000 ft (maximum)
Attenuation.	0.35 dB/100 ft @ 125 kHz (maximum) 0.70 dB/100 ft @ 500 kHz (maximum) 0.95 dB/100 ft @ 1.00 MHz (maximum)

**Table 8-3.51 Cable VI: DC Power Pair Specification**

Physical Characteristics	Specification
Conductor pair size	#26 AWG Copper (minimum); 19 strands minimum (individually tinned)
Insulation diameter	.041 inches (minimum)
Colors	Red Black
Pair Twist	5/ft (approximately)
Tape shield over pair	1 mil/ 1 mil, AL/Mylar
Electrical Characteristics	Specification
DCR - @ 20 deg C	44.4 Ohms/1000 ft (maximum)

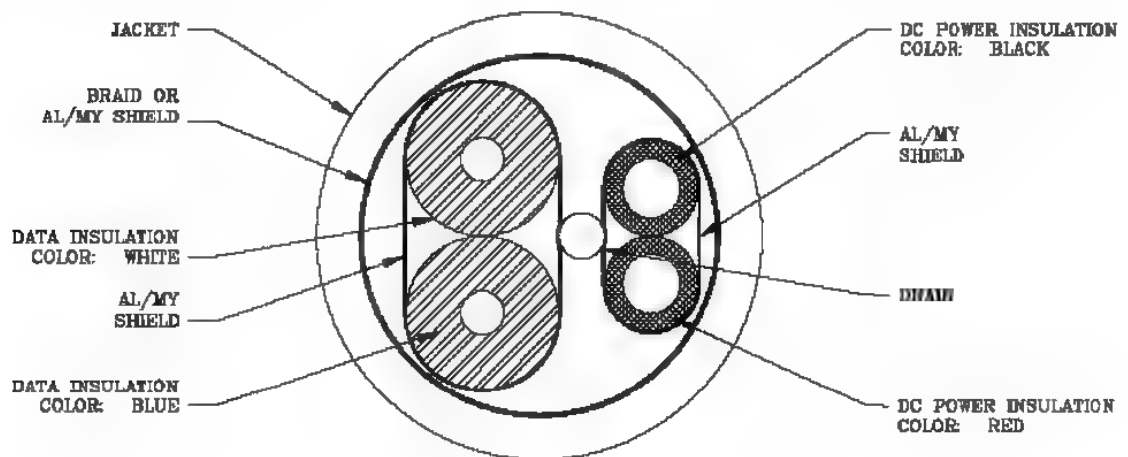
**Table 8-3.52 Cable VI: General Specification**

Physical Characteristics	Specification
Geometry	Two shielded pairs, Common axis with drain wire in center
Over braid shield	Tinned copper stranded (38 AWG min.) shield with 65% minimum coverage OR overall tape shield (1 mil/1mil, AL/Mylar)
Drain wire	#26 AWG Copper (minimum); 19 strands minimum (individually tinned)
Outside diameter	Specified by vendor
Roundness	Radius delta to be within 20% of 0.5 O.D.
Jacket marking	Vendor name and part # and additional markings
Electrical Characteristics	Specification
DCR (braid+tape+drain) @ 20 C	6.0 Ohms/1000 ft (maximum)
Applicable Environmental Characteristics	Specification
Agency Certification (U.S. and Canada)	Compliant with local government regulations
Flexure	Specified by vendor
Bend Radius	Specified by vendor
Operating ambient temperature	-20 to +70 C @ 1.68 amps; de-rate current linearly to zero at 80 C (minimum)
Storage temperature	-40 to +85 C (minimum)
Pull tension	20 lbs max.
Connector Compatibility	Mini, Micro, Open, M8
Topology Compatibility	Trunk, Drop

**Table 8-3.53 Cable VI: Topology**

Data Rate	Max Cable Distance	Trunk Exchange (Thick cable)	Cumulative Drop	Maximum Drop
125kb	75 m (246 ft)	7.6	156m (512 ft)	6m (20 ft)
250kb	66 m (216 ft)	3.8	78m (256 ft)	6m (20 ft)
500kb	50 m (164 ft)	1.5	39m (128 ft)	6m (20 ft)

**Figure 8-3.28 Cable VI: Physical Configuration**

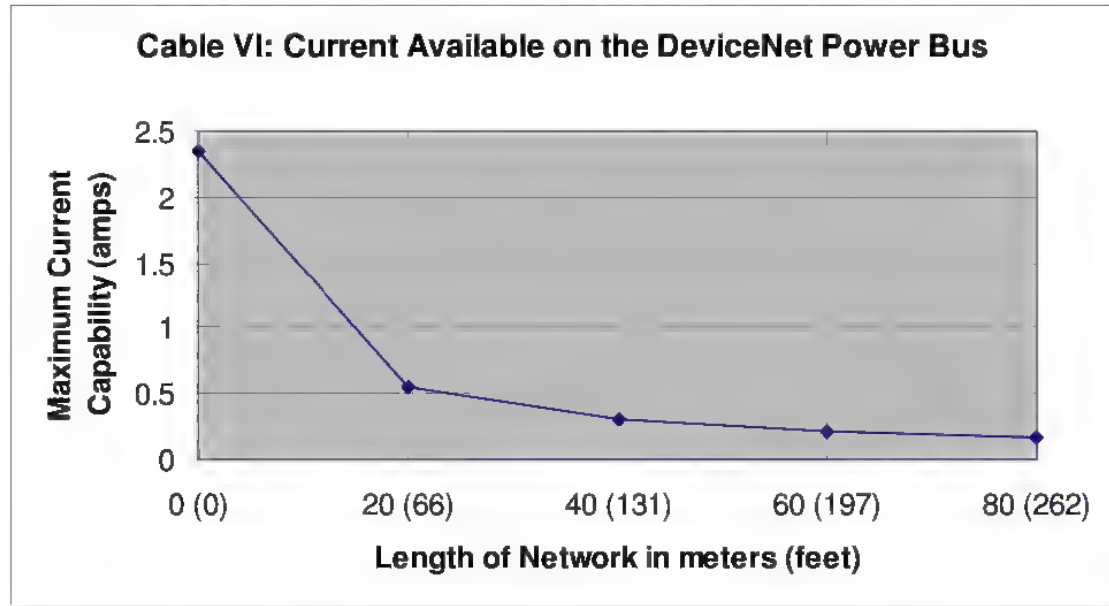




**Table 8-3.54 Cable VI: Max. Current Available (amps) Based on Network Length**

Network Length in meters (feet)	0	20 (66)	40 (131)	60 (197)	80 (262)
Maximum Current in amps	2.34	0.55	0.31	0.22	0.17

**Figure 8-3.29 Cable VI: Current Available on the DeviceNet Power Bus**



## 8-3.9 DeviceNet Tap Specifications

The following tap profiles are more detailed than the specifications found in the Physical Layer Requirements chapter. Included in this section are profiles for both sealed and open taps that are supported by DeviceNet.

### 8-3.9.1 Tap Profile Template

A tap profile defines the General Tap Specifications, Internal Pass Through, Conductor Specifications, Internal Drop Conductor Specifications, Electrical Specifications and Environmental Specifications. The following table defines the minimum fields that must be defined for a DeviceNet tap profile.

**Table 8-3.55 yyy Trunk to xxx Drop Profile**

Internal Pass Through Conductor Description	Specification
Drain wire conductor	
Length of conductor	<#> inches (maximum)
Conductor resistance	<#> mOhms (maximum)
Conductor equivalent	<size> (minimum)
Power conductors	
Length of conductor	<#> inches (maximum)
Conductor resistance	<#> mOhms (maximum)
Conductor equivalent	<size> (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	<#> inches (maximum)
Conductor resistance	<#> mOhms (maximum)
Conductor equivalent	<size> (minimum)
Internal Drop Conductor Description	Specification
Drain wire conductor	
Length of conductor	<#> inches (maximum)
Conductor resistance	<#> mOhms (maximum)
Conductor equivalent	<size> (minimum)
Power conductors	
Length of conductor	<#> inches (maximum)
Conductor resistance	<#> mOhms (maximum)
Conductor equivalent	<size> (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	<#> inches (maximum)
Conductor resistance	<#> mOhms (maximum)
Conductor equivalent	<size> (minimum)

Electrical Characteristics	Specification
Operating Voltage	<#> volt (minimum)
Trunk Contact Rating	<#> amps (minimum)
Drop Contact Rating	<#> amps (minimum)
Tap Capacitance	<#> pF (maximum)
Number of Taps	<#> per network (maximum)
De-rating	
125kb	<#>m of trunk or cumulative drop per tap over <#> taps
250kb	<#>m of trunk or cumulative drop per tap over <#> taps.
500kb	<#>m of trunk or cumulative drop per tap over <#> taps.
Environmental Characteristics	Specification
Water resistance	IP<#> and NEMA <#>
Oil resistance	
Operating ambient temperature	<#> to <#>C
Storage temperature	<#> to <#>C
Marking/Labeling	Vendor name
Recommend Keying	

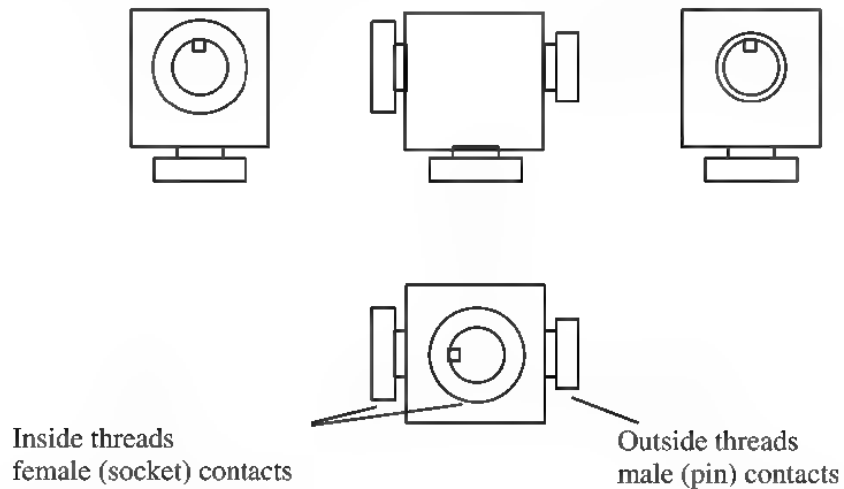
### 8-3.9.2 Mini Trunk to Mini Drop Tap Profile

**Table 8-3.56 Mini Trunk to Mini Drop Profile**

Internal Pass Through Conductor Description	Specification
Drain wire conductor	
Length of conductor	4 inches (maximum)
Conductor resistance	2.3 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Power conductors	
Length of conductor	4 inches (maximum)
Conductor resistance	1.2 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	4 inches (maximum)
Conductor resistance	2.3 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Internal Drop Conductor Description	Specification
Drain wire conductor	
Length of conductor	7 inches (maximum)
Conductor resistance	10 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Power conductors	
Length of conductor	7 inches (maximum)
Conductor resistance	4 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	7 inches (maximum)
Conductor resistance	N/A
Conductor equivalent	24 AWG (minimum)
Electrical Characteristics	Specification
Operating Voltage	25 volt (minimum)
Trunk Contact Rating	8 amps (minimum)
Drop Contact Rating	8 amps (minimum)
Tap Capacitance	15 pF (maximum)
Number of Taps	70 per network (maximum)
De-rating	
125kb	None
250kb	None
500kb	1m of trunk or cumulative drop per tap over 39 taps

Environmental Characteristics	Specification
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL 1277, OIL RES II
Operating ambient temperature	-40 to +70C @ 8 amps; de-rate current linearly to 0 @ 80C
Storage temperature	-40 to +85C
Marking/Labeling	Vendor name and additional markings
Recommend Keying	See Figure 8-3.30

**Figure 8-3.30 Mini Trunk to Mini Drop Tap Recommended Keying**



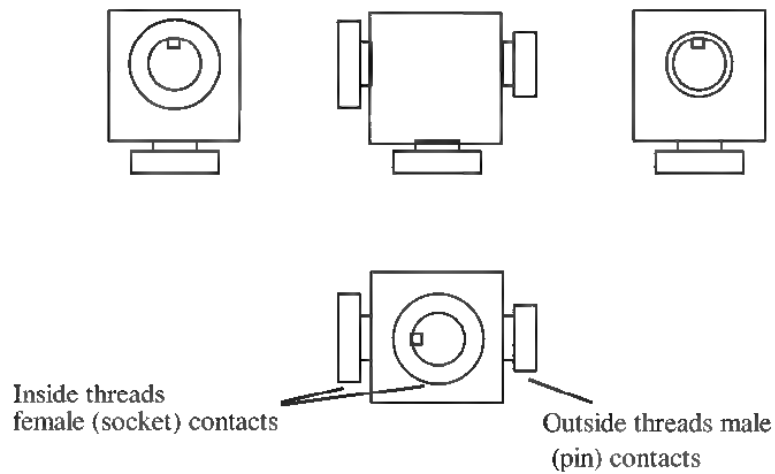
### 8-3.9.3 Micro Trunk to Micro Drop Tap Profile

**Table 8-3.57 Micro Trunk to Micro Drop Tap Profile**

Internal Pass Through Conductor Description	Specification
Drain wire conductor	
Length of conductor	4 inches (maximum)
Conductor resistance	2.3 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Power conductors	
Length of conductor	4 inches (maximum)
Conductor resistance	1.2 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	4 inches (maximum)
Conductor resistance	2.3 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Internal Drop Conductor Description	Specification
Drain wire conductor	
Length of conductor	7 inches (maximum)
Conductor resistance	10 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Power conductors	
Length of conductor	7 inches (maximum)
Conductor resistance	10 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	7 inches (maximum)
Conductor resistance	N/A
Conductor equivalent	24 AWG (minimum)
Electrical Characteristics	Specification
Operating Voltage	25 volt (minimum)
Trunk Contact Rating	3 amps (minimum)
Drop Contact Rating	3 amps (minimum)
Tap Capacitance	15 pF (maximum)
Number of Taps	70 per network (maximum)
De-rating	
125kb	None
250kb	None
500kb	1m of trunk or cumulative drop per tap over 39 taps.

Environmental Characteristics	Specification
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL 1277, OIL RES II
Operating ambient temperature	-40 to +70C @ 8 amps; de-rate current linearly to 0 @ 80C
Storage temperature	-40 to +85C
Marking/Labeling	Vendor name and additional markings
Recommend Keying	See Figure 8-3.31

**Figure 8-3.31 Micro Trunk to Micro Drop Tap Recommended Keying**



#### 8-3.9.4 Mini Trunk to Micro Drop Tap Profile

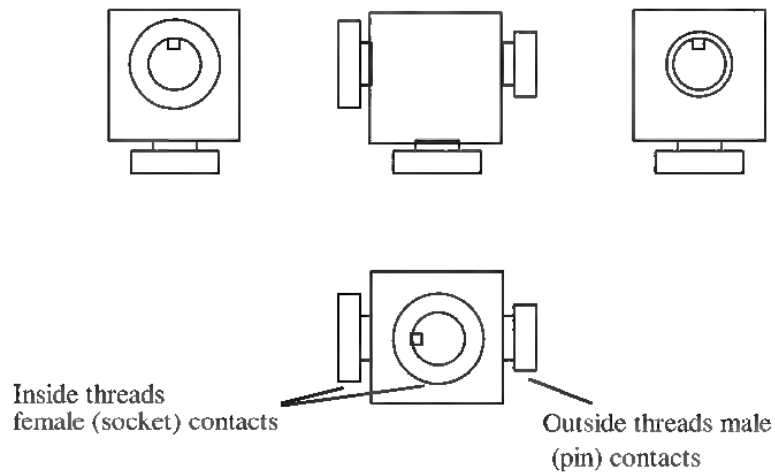
**Table 8-3.58 Mini Trunk to Micro Drop Profile**

Internal Pass Through Conductor Description	Specification
Drain wire conductor	
Length of conductor	4 inches (maximum)
Conductor resistance	2.3 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Power conductors	
Length of conductor	4 inches (maximum)
Conductor resistance	1.2 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	4 inches (maximum)
Conductor resistance	2.3 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Internal Drop Conductor Description	Specification
Drain wire conductor	
Length of conductor	7 inches (maximum)
Conductor resistance	10 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Power conductors	
Length of conductor	7 inches (maximum)
Conductor resistance	10 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	7 inches (maximum)
Conductor resistance	N/A
Conductor equivalent	24 AWG (minimum)
Electrical Characteristics	Specification
Operating Voltage	25 volt (minimum)
Trunk Contact Rating	8 amps (minimum)
Drop Contact Rating	3 amps (minimum)
Tap Capacitance	15 pF (maximum)
Number of Taps	70 per network (maximum)
De-rating	
125kb	None
250kb	None
500kb	1m of trunk or cumulative drop per tap over 39 taps.



Environmental Characteristics	Specification
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL 1277, OIL RES II
Operating ambient temperature	-40 to +70C @ 8 amps; de-rate current linearly to 0 @ 80C
Storage temperature	-40 to +85C
Marking/Labeling	Vendor name and additional markings
Recommend Keying	See Figure 8-3.32

**Figure 8-3.32 Mini Trunk to Micro Drop Tap Recommended Keying**



### 8-3.9.5 Flat Trunk to Micro Drop Tap Profile

**Table 8-3.59 Flat Trunk to Micro Drop Profile**

Internal Pass Through Conductor Description	Specification
Drain wire conductor	
Length of conductor	N/A
Conductor resistance	N/A
Conductor equivalent	N/A
Power conductors	
Length of conductor	4 inches (maximum)
Conductor resistance	1.2 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	4 inches (maximum)
Conductor resistance	2.3 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Internal Drop Conductor Description	Specification
Drain wire conductor	
Length of conductor	N/A
Conductor resistance	N/A
Conductor equivalent	N/A
Power conductors	
Length of conductor	7 inches (maximum)
Conductor resistance	10 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	7 inches (maximum)
Conductor resistance	N/A
Conductor equivalent	24 AWG (minimum)
Electrical Characteristics	Specification
Operating Voltage	25 volt (minimum)
Trunk Contact Rating	8 amps (minimum)
Drop Contact Rating	3 amps (minimum)
Tap Capacitance	10 pF (maximum)
Number of Taps	70 per network (maximum)
De-rating	
125kb	None
250kb	None
500kb	None

<b>Environmental Characteristics</b>	<b>Specification</b>
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL 1277, OIL RES II
Operating ambient temperature	-40 to +70C @ 3 amps; de-rate current linearly to 0 @ 80C
Storage temperature	-40 to +85C
Marking/Labeling	Vendor name and additional markings
Recommend Keying	None

### 8-3.9.6 Flat Trunk to Mini Drop Tap Profile

**Table 8-3.60 Flat Trunk to Mini Drop Profile**

Internal Pass Through Conductor Description	Specification
Drain wire conductor	
Length of conductor	N/A
Conductor resistance	N/A
Conductor equivalent	N/A
Power conductors	
Length of conductor	4 inches (maximum)
Conductor resistance	1.2 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	4 inches (maximum)
Conductor resistance	2.3 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Internal Drop Conductor Description	Specification
Drain wire conductor	
Length of conductor	N/A
Conductor resistance	N/A
Conductor equivalent	N/A
Power conductors	
Length of conductor	7 inches (maximum)
Conductor resistance	10 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	7 inches (maximum)
Conductor resistance	N/A
Conductor equivalent	24 AWG (minimum)
Electrical Characteristics	Specification
Operating Voltage	25 volt (minimum)
Trunk Contact Rating	8 amps (minimum)
Drop Contact Rating	8 amps (minimum)
Tap Capacitance	10 pF (maximum)
Number of Taps	70 per network (maximum)
De-rating	
125kb	None
250kb	None
500kb	None

<b>Environmental Characteristics</b>	<b>Specification</b>
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL 1277, OIL RES II
Operating ambient temperature	-40 to +70C @ 8 amps; de-rate current linearly to 0 @ 80C
Storage temperature	-40 to +85C
Marking/Labeling	Vendor name and additional markings
Recommend Keying	None

### 8-3.9.7 Flat Trunk to Open Drop Tap Profile

Table 8-3.61 Flat Trunk to Open Drop Profile

Internal Pass Through Conductor Description	Specification
Drain wire conductor	
Length of conductor	N/A
Conductor resistance	N/A
Conductor equivalent	N/A
Power conductors	
Length of conductor	4 inches (maximum)
Conductor resistance	1.2 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	4 inches (maximum)
Conductor resistance	2.3 mOhms (maximum)
Conductor equivalent	22 AWG (minimum)
Internal Drop Conductor Description	Specification
Drain wire conductor	
Length of conductor	N/A
Conductor resistance	N/A
Conductor equivalent	N/A
Power conductors	
Length of conductor	7 inches (maximum)
Conductor resistance	10 mOhms (maximum)
Conductor equivalent	18 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	7 inches (maximum)
Conductor resistance	N/A
Conductor equivalent	24 AWG (minimum)
Electrical Characteristics	Specification
Operating Voltage	25 volt (minimum)
Trunk Contact Rating	8 amps (minimum)
Drop Contact Rating	8 amps (minimum)
Tap Capacitance	10 pF (maximum)
Number of Taps	70 per network (maximum)
De-rating	
125kb	None
250kb	None
500kb	None

<b>Environmental Characteristics</b>	<b>Specification</b>
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL 1277, OIL RES II
Operating ambient temperature	-40 to +70C @ 8 amps; de-rate current linearly to 0 @ 80C
Storage temperature	-40 to +85C
Marking/Labeling	Vendor name and additional markings
Recommend Keying	None

### 8-3.9.8 Micro Trunk to M8 Drop Tap Profile

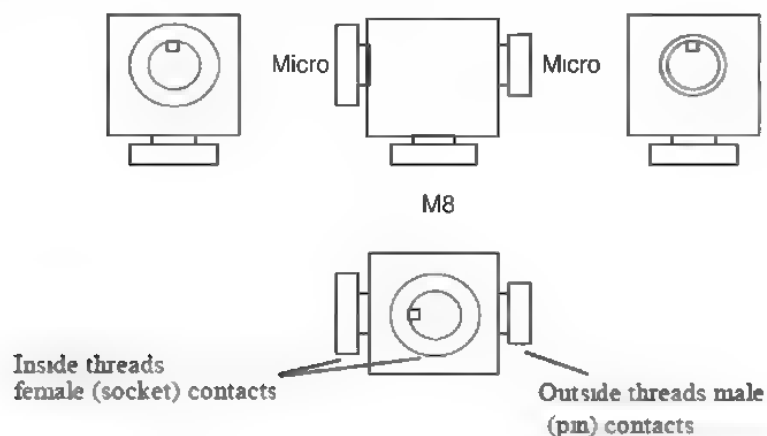
Table 8-3.62 Micro Trunk to M8 Drop Tap Profile

Internal Pass Through Conductor Description	Specification
Drain Wire conductor	
Length of conductor	4 inches (maximum)
Conductor resistance	9.3 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Power conductors	
Length of conductor	4 inches (maximum)
Conductor resistance	9.3 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	4 inches (maximum)
Conductor resistance	9.3 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Internal Drop Conductor Description	Specification
Drain Wire conductor	
Length of conductor	7 inches (maximum)
Conductor resistance	16.2 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Power conductors	
Length of conductor	7 inches (maximum)
Conductor resistance	16.2 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	7 inches (maximum)
Conductor resistance	N/A
Conductor equivalent	24 AWG (minimum)
Electrical Characteristics	Specification
Operating Voltage	25 Volt (minimum)
Trunk Contact Rating	3 amps (minimum)
Drop Contact Rating	3 amps (minimum)
Capacitance between conductors	15 pF (maximum)
Number of Taps	70 per network (maximum)
De-rating	
125kb	None
250kb	None
500kb	1m of trunk or cumulative drop per tap over 39 taps



Environmental Characteristics	Specification
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL 1277, OIL RES II
Operating ambient temperature	-40 to +70C @ 3 amps, de-rate current linearly to 0 @ 80C
Storage temperature	-40 to +85C
Marking/Labeling	Vendor name and additional markings
Recommend Keying	See Figure 8-3.33

**Figure 8-3.33 Micro Trunk to M8 Drop Tap Recommend Keying**



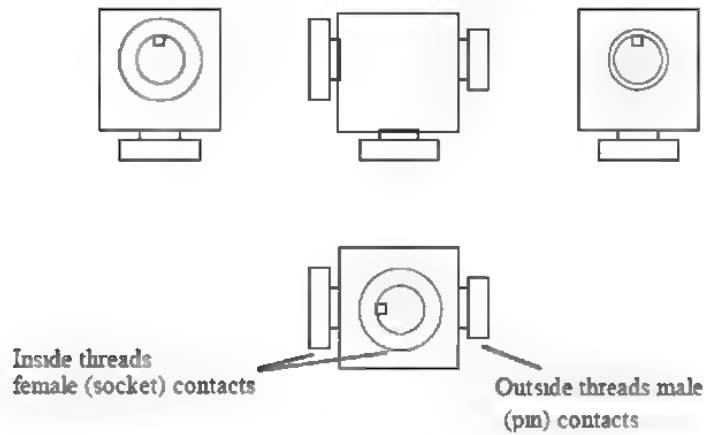
### 8-3.9.9 M8 Trunk to M8 Drop Tap Profile

Table 8-3.63 M8 Trunk to M8 Drop Tap Profile

Internal Pass Through Conductor Description	Specification
Drain Wire conductor	
Length of conductor	4 inches (maximum)
Conductor resistance	9.3 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Power conductors	
Length of conductor	4 inches (maximum)
Conductor resistance	9.3 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	4 inches (maximum)
Conductor resistance	9.3 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Internal Drop Conductor Description	Specification
Drain Wire conductor	
Length of conductor	7 inches (maximum)
Conductor resistance	16.2 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Power conductors	
Length of conductor	7 inches (maximum)
Conductor resistance	16.2 mOhms (maximum)
Conductor equivalent	24 AWG (minimum)
Signal conductors	
Impedance	120 Ohms
Length of conductor	7 inches (maximum)
Conductor resistance	N/A
Conductor equivalent	24 AWG (minimum)
Electrical Characteristics	Specification
Operating Voltage	25 Volt (minimum)
Trunk Contact Rating	3 amps (minimum)
Drop Contact Rating	3 amps (minimum)
Tap Capacitance	15 pF (maximum)
Number of Taps	70 per network (maximum)
De-rating	
125kb	None
250kb	None
500kb	1m of trunk or cumulative drop per tap over 39 taps
Environmental Characteristics	Specification
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL 1277, OIL RES II

Internal Pass Through Conductor Description	Specification
Operating ambient temperature	-40 to +70C @ 3 amps; de-rate current linearly to 0 @ 80C
Storage temperature	-40 to +85C
Marking/Labeling	Vendor name and additional markings
Recommend Keying	See Figure 8-3.34

**Figure 8-3.34 M8 Trunk to M8 Drop Tap Recommend Keying**



### **8-3.10 Power Tap Profiles**

The following specification includes general descriptions for both sealed and open power taps that are supported by DeviceNet. These specifications are more detailed than the ones found in the Physical Layer Requirements chapter.

#### **8-3.10.1 Power Tap Specifications**

Included are the following tables regarding DeviceNet Power Taps:

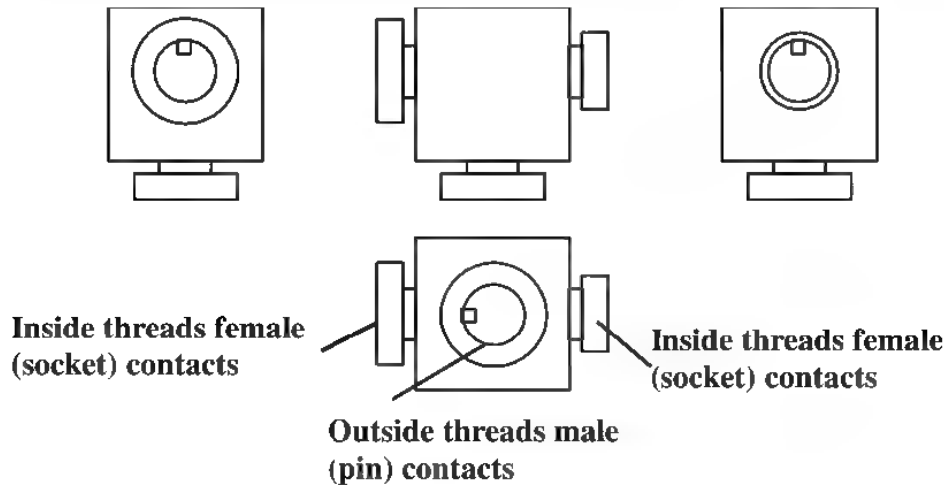
- General Power Tap Specifications
- Internal Pass Through Conductor Specifications
- Internal Power Drop Conductor Specifications
- Electrical Specifications
- Environmental Specifications

**Table 8-3.64 Recommended PowerTap Specifications**

Physical Characteristic	Specification
Marking/Labeling	Power Tap, Vendor name & part #
Trunk connectors	Sealed or open per DeviceNet connector specifications
Power Drop connector	Sealed or open per DeviceNet connector specifications except using 4 pin version
Protection circuit (times 2) (optional)	Non-auto reset fuse or circuit breaker in power tap
Protection serviceability (optional)	Field resettable/replaceable
Shield ground connection	10-32 screw or stud on case or grounding wire.

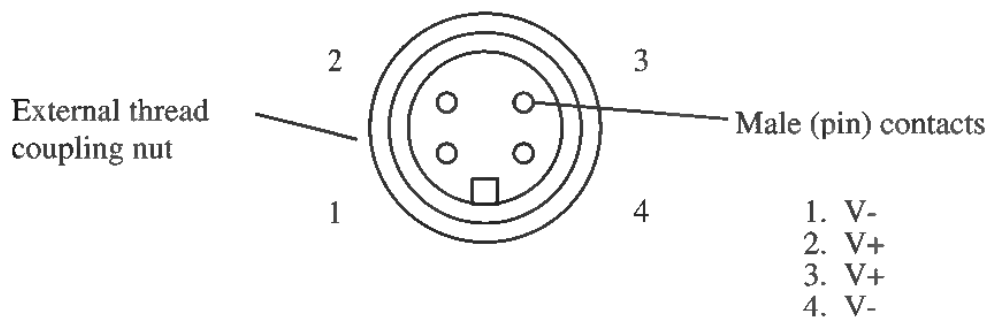
### 8-3.10.2 Recommended Mini Connector Gender and Recommended Keying

Figure 8-3.35 Mini Connector - Recommended Keying



### 8-3.10.3 Recommended Mini Connector Pin Out for Drop to Power Supply

Figure 8-3.36 Mini Connector Pin Out for Drop to Power Supply



**Note:** Recommended connector gender, prevents any exposed powered male contacts with single power supply networks. Certain applications might require different connector gender options than shown above, to minimize exposed power pins.

When adding or relocating power supplies on an existing network between male-female connection cordsets, one of the trunk connectors may be a male connector as specified in section 8-3.12.

For replacing an existing power tap, the power tap drop connector may be a female connector.

#### 8-3.10.4 Typical Power Tap Schematic

Figure 8-3.37 Schematic of a Typical Power Tap

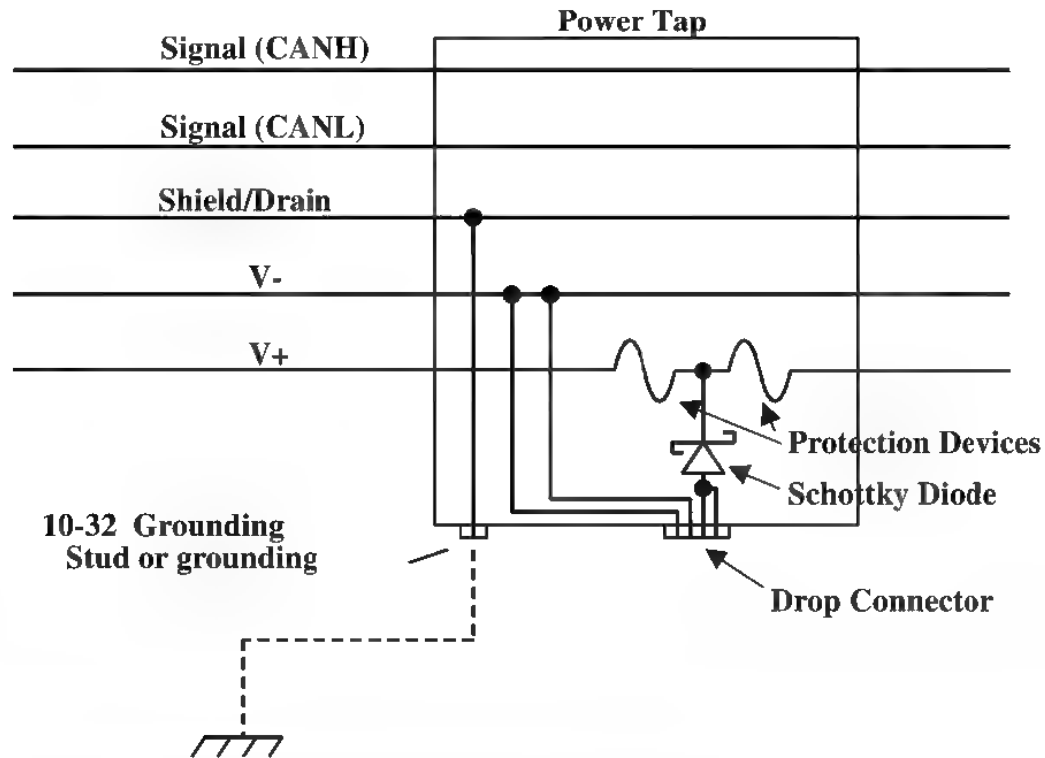


Table 8-3.65 Internal Pass Through Conductor Specifications

Conductor Description	Specification
Drain wire conductor (stranded or solid)	7 inches maximum of unshielded conductor. Maximum conductor resistance 4 mOhm. (Equivalent to 7" of 18 AWG). Minimum conductor equivalent at any point 18 AWG.
V- Power conductor (stranded or solid)	7 inches maximum conductor length. Maximum conductor resistance of 2.1 mOhms. (Equivalent to 7" of 15 AWG). Minimum conductor equivalent 16 AWG.
CANH and CANL Signal conductors (stranded or solid)	7 inches maximum conductor length. Maximum resistance of 4.1 mOhms. (Equivalent to 7" of 18 AWG). Minimum conductor equivalent at any point 22 AWG.

Table 8-3.66 Internal Power Drop Conductor Specifications

Conductor Description	Specification
Drain wire conductor to grounding terminal (stranded or solid)	Maximum conductor length of 7 inches. Maximum resistance 2.1 mOhms. (Equivalent to 7" of 15 AWG). Minimum allowable conductor at any point 16 AWG.
V+ and V- Power conductors (stranded or solid)	12 inches maximum conductor length per leg. Maximum resistance per leg (excluding protection) 3.6 mOhms. (Equivalent to 12" of 15 AWG). Minimum conductor equivalent at any location 16 AWG. Any conductor carrying between 8 and 16 amps must be 12 AWG equivalent or larger.

**Table 8-3.67 Electrical Specifications**

Electrical Characteristics	Specification
Protection Circuit (optional)	Hold, 8 amps for thick trunk, 3 amps for thin trunk. 1.5 sec min. to 120 sec max trip time for 20 amps at 25 deg C. Slo-Blo or Normal-Blo Allowable
Schottky Diode (optional)	20 amp, 30 volt minimum continuous capability over environment. Max Vf of .73 volts @ Tc=125 C and 16 A minimum test condition. MBR3045 Recommended
Operating Voltage	25 volt minimum
Contact Rating	Trunk line: 8 amps minimum for thick trunk, 3 amps minimum for thin trunk, over temperature for all contacts. Drop line: The sum of the contact ratings for both V+ and V- must be 16 amps minimum for thick drop and 6 amps minimum for thin drop over temperature

**Table 8-3.68 Environmental Specifications**

Applicable Environmental Characteristics	Specification
Water resistance (sealed)	IP67 and NEMA 4, 6, 6P, 13 for sealed applications.
Oil resistance (sealed)	UL-1277, OIL RES II for sealed applications.
Operating ambient temperature	-40 to +70 C with maximum continuous power on all conductors. De-rate linearly to 0 amps at 80 deg C.
Storage temperature	-40 to +85 C

Note: The use of manufacturer's names and catalog numbers may occur in these specifications. These occurrences do not imply or constitute qualification. These occurrences are intended as a temporary measure and will exist only until suitable replacement standards are developed.

### **8-3.11 DeviceNet Power Specifications**

Follow these specifications when designing your own power supply or establishing the power delivery aspects of the network including the device voltage regulator. Included are the following tables:

- Power Supply Specifications
- Network Voltage Tolerance Design Stack Up
- Network Voltage Drop Budget
- Schottky Diode Specifications
- DC/DC Converter

#### **8-3.11.1 Power Supply Specifications**

The following table outlines the requirements of a power supply that is to be connected to DeviceNet. These specifications apply to single supply, multiple supply and also to nodes that are going to also provide power to the network.

**Table 8-3.69 DeviceNet Power Supply Specifications**

Specification	Parameter
Initial Tolerance	24 volts+/- 1% or adjustable to 0.2%
Line Regulation	0.3% max
Load Regulation	0.3% max
Temperature Coefficient	0.03% per deg C max
Output Ripple	250 mV p-p
Load Capacitance Capability	7000 uF max
Temperature Range	Operating*0 to 60 deg C Non-operating -40 to 85 deg C *De-rating acceptable for 60 deg C operation
Inrush Current Limit	less than 65 Amps peak
Over Voltage Protection	yes (no value specified)
Over Current Protection	yes (current limit 125 % max)
Turn-on Time (with full load)	250 msec max/5% of final value
Turn-on Overshoot	0.2% max
Stability	0 to 100 % load (all conditions)
Isolation	output isolated from AC and Chassis ground
Output Voltage	24 Volts +/- 1 %
Output current	up to 16 A continuous
Surge current capability	10 % reserve capability



### 8-3.11.2 Network Voltage Tolerance Stack Up

The following table outlines the stack up of tolerances leading to the +/- 4% required by DeviceNet. By using the information provided, performance trade-offs can be made on the power supply and Schottky diode specifications and still meet DeviceNet requirements. Maximum tolerance for the DeviceNet system is 24 volts +/- 4.0%.

**Table 8-3.70 Recommended Tolerance Stack Up for DeviceNet**

Specification	Parameter
Initial Setting	1.0%
Line Regulation	0.3%
Load Regulation	0.3%
Temperature Coefficient*	0.6%
Schottky Diode Drop (0.65 V nominal)	0.75% (of 24 volts)
Time Drift	1.05%
Total Stack Up	4.0%

\*The temperature coefficient tolerance of 0.6% is based on an actual rating of .03% per deg C and a 20 deg C differential between supplies that are used on the bus. If a supply in one location is in an ambient of 40 deg C, it is assumed that other supplies are within 10 deg C or in the range of 30 to 50 deg C (or another 20 deg C range). If this stipulation is not met, and all the other tolerances are just being met, then power capability will need to be de-rated.

### 8-3.11.3 Network Voltage Drop Budget

The specifications in Table 8-3.71 indicate actual voltage levels and the maximum tolerances provided for power configuration on DeviceNet.

**Table 8-3.71 Voltage Drop Budget**

Specification	Tolerance	Actual Voltage
Power Supply Tolerance	+/-1%	0.24 V
Total Temperature Drift	*1.8%	0.432 V
Line Regulation	0.3%	0.072 V
Load Regulation	0.3%	0.072 V
Diode Drop	3.04%	0.73 V
Common Mode Drop	41.66%	10.0 V
Supply Ripple	0.3125%	0.075 V
Input Ripple	3.125%	0.75 V
Total Voltage Budget	51.5%	12.96 V

\*Based on a temperature coefficient of .03% and an ambient temperature of 60 deg C.

### 8-3.11.4 Schottky Diode Specification

The following specifications define the requirements of the Schottky diode, if used to connect power supplies to DeviceNet.

**Table 8-3.72 Recommended Schottky Diode Specifications**

Specification	Parameter
Voltage Rating	30 V min
Current Rating	25 amps
Max Vf @Ifm =I0	0.73V @ Tc=125 deg C with 16 A

### 8-3.11.5 Node Regulator Specification

**Table 8-3.73 DC/DC Converter Specifications**

Specification	Parameter
Input Voltage Range	11 volts to 25 volts
Efficiency	70% minimum
Isolation (if required)	500 volts
Turn on delay	Linear Regulators none Switchers; <100mA 2 to 10 msec .1A - .5A 5 - 15 msec .5A - 1A 10 - 20 msec 1A - 2A 15 - 30 msec >2A 20 - 40 msec
Output short circuit protection	current limit
Reverse polarity protection	Schottky diode in gnd path

## 8-3.12 Connectors

The following connector profiles are more detailed than the specifications found in the Physical Layer Requirements chapter. Included in this section are profiles for both sealed and open connectors that are supported by DeviceNet.

### 8-3.12.1 Connector Profile Template

A connector profile defines the Male and Female General Specifications, Contact Specifications, Electrical Specifications and Environmental Specifications. The following table defines the minimum fields that must be defined for a DeviceNet connector profile.

**Table 8-3.74 Connector Profile Template**

Male General Characteristics	Specification
Number of Pins	<#>
Coupling Nut	<Male/Female/None>
Coupling Nut Thread	
Rotation	<Required/Optional>
Standard	
Pinout	Drain. Pin #, V+ : Pin #, V- : Pin #, CAN H : Pin #, CAN L : Pin #
Female General Characteristics	Specification
Number of Sockets	<#>
Coupling Nut	<Male/Female/None>
Coupling Nut Thread	
Rotation	<Required/Optional>
Standard	
Pinout	Drain. Pin #, V+ : Pin #, V- : Pin #, CAN H : Pin #, CAN L : Pin #
Physical Characteristics	Specification
Wiping Contact Plating Requirements	
Wiping Contact Life	<#> insertion-extractions.
Electrical Characteristics	Specification
Operating Voltage	<#> volt minimum
Contact Rating	<#> amps minimum
Contact Resistance	Nominal: Less than <#> mOhm. Maximum: <#> mOhms over life.
Environmental Characteristics	Specification
Water resistance	IP<#> and NEMA <#>
Oil resistance	
Operating ambient temperature	<#> to <#> C
Storage temperature	<#> to <#> C

### 8-3.12.2 Open Connector Profile

**Table 8-3.75 Open Connector**

Male General Characteristics	Specification
Number of Pins	5
Coupling Nut	None
Coupling Nut Thread	None
Rotation	None
Standard	See Figure 8-3.38 for interface geometry.
Pinout	V- : Pin 1, CAN_L : Pin 2, Drain : Pin 3, CAN_H : Pin 4, V+ : Pin 5
Female General Characteristics	Specification
Number of Sockets	5
Coupling Nut	None
Coupling Nut Thread	None
Rotation	None
Standard	See Figure 8-3.39 for interface geometry.
Pinout	V- : Pin 1, CAN_L : Pin 2, Drain : Pin 3, CAN_H : Pin 4, V+ : Pin 5
Physical Characteristics	Specification
Wiping Contact Plating Requirements	30 micro inch gold minimum over 50 micro inch nickel minimum or 5 micro inch gold minimum over 20 micro inch palladium-nickel minimum over 50 micro inch nickel. All gold must be 24 Karat
Wiping Contact Life	100 insertion-extractions.
Electrical Characteristics	Specification
Operating Voltage	25 volt minimum
Contact Rating	8 amps minimum
Contact Resistance	Nominal. Less than 1 mOhm. Maximum: 5 mOhms over life.
Environmental Characteristics	Specification
Water resistance	None
Oil resistance	None
Operating ambient temperature	-40 to +70 deg C full power with linear derating to 0 amps at 80 deg C.
Storage temperature	-40 to +85 deg C

Figure 8-3.38 Open Connector Pinout

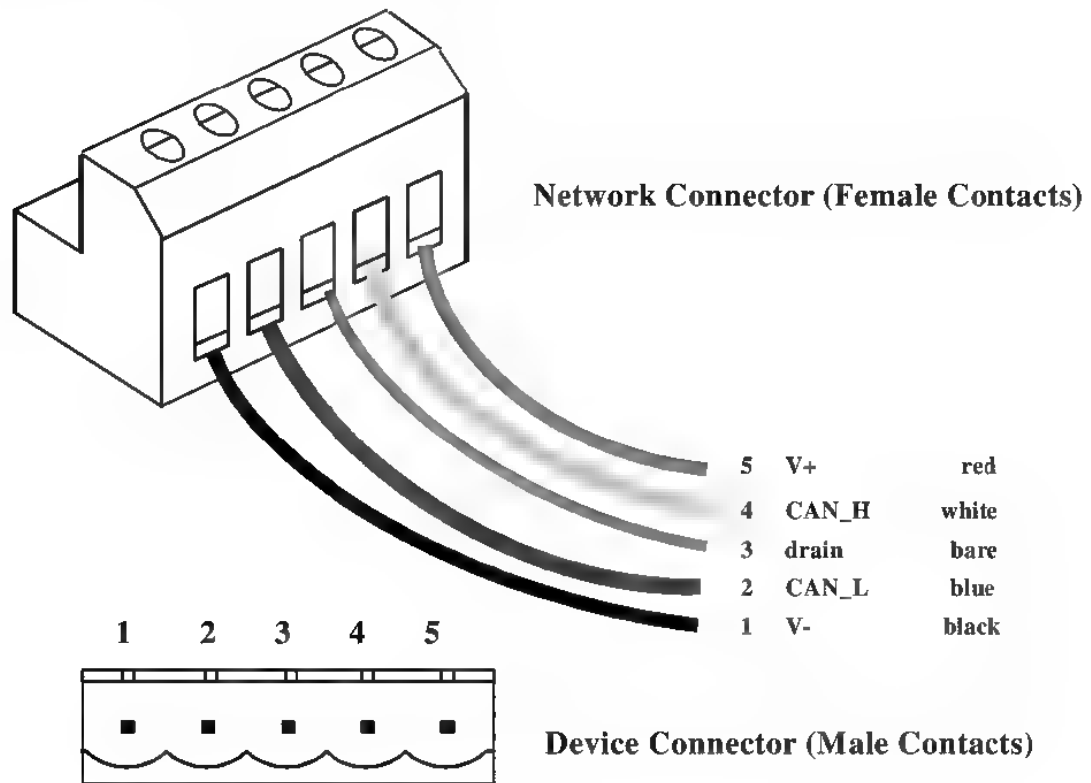
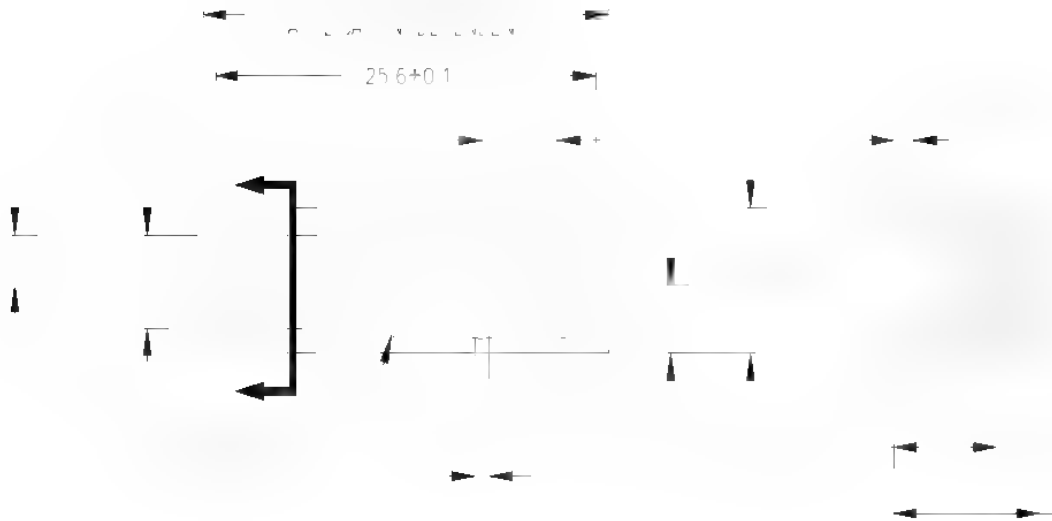


Figure 8-3.39 Open Connector Geometry



### 8-3.12.3 Sealed Mini Connector Profile

Table 8-3.76 Sealed Mini Connector

Male General Characteristics	Specification
Number of Pins	5
Coupling Nut	Male
Coupling Nut Thread	7/8-16 UN-2A THD
Rotation <sup>1</sup>	Optional
Standard	ANSI/B93.55M-1981 intermateability requirements
Pinout	Drain : Pin 1, V+ : Pin 2, V- : Pin 3, CAN_H : Pin 4, CAN_L : Pin 5
Female General Characteristics	Specification
Number of Sockets	5
Coupling Nut	Female
Coupling Nut Thread	7/8-16 UN-2B THD
Rotation <sup>1</sup>	Required
Standard	ANSI/B93.55M-1981 intermateability requirements
Pinout	Drain : Pin 1, V+ : Pin 2, V- : Pin 3, CAN_H : Pin 4, CAN_L : Pin 5
Physical Characteristics	Specification
Wiping Contact Plating Requirements	30 micro inch gold minimum over 50 micro inch nickel minimum or 5 micro inch gold minimum over 20 micro inch palladium-nickel minimum over 50 micro inch nickel. All gold must be 24 Karat
Wiping Contact Life	1000 insertion-extractions.
Electrical Characteristics	Specification
Operating Voltage	25 volt minimum
Contact Rating	8 amps minimum
Contact Resistance	Nominal: Less than 1 mOhm. Maximum: 5 mOhms over life.
Environmental Characteristics	Specification
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL-1277, OIL RES II
Operating ambient temperature	-40 to +70 deg C full power with linear derating to 0 amps at 80 deg C.
Storage temperature	-40 to +85 deg C
<sup>1</sup> This requirement only applies to connectors on cable assemblies/cordsets	

Figure 8-3.40 Mini Connector Pinout

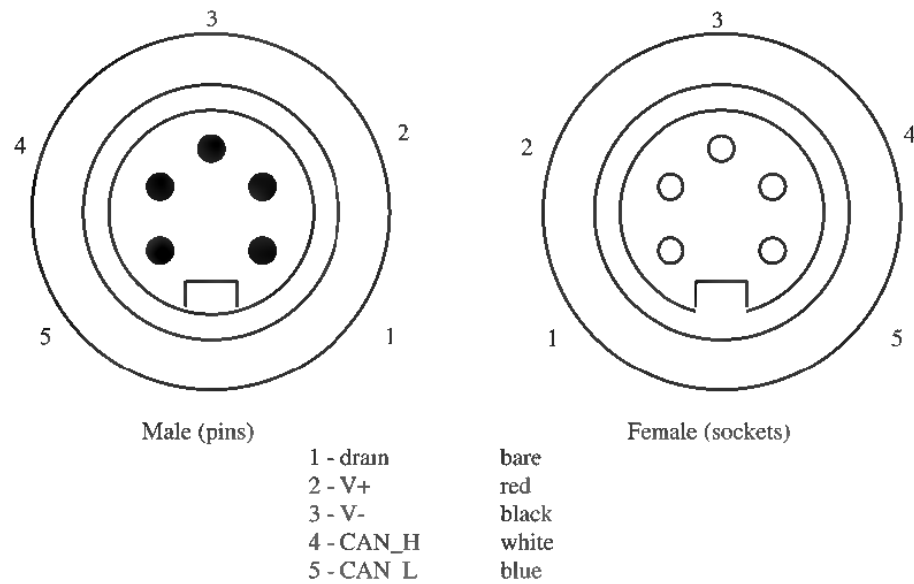


Figure 8-3.41 Mini 90 Degree Connector





#### 8-3.12.4 Sealed Micro Connector Profile

**Table 8-3.77 Sealed Micro Connector**

Male General Characteristics	Specification
Number of Pins	5
Coupling Nut	Male
Coupling Nut Thread	M12 x 1 6G fit
Rotation <sup>1</sup>	Optional
Standard	IEC 60947-5-2, Figure D.2 intermateability requirements
Pinout	Drain : Pin 1, V+ : Pin 2, V- : Pin 3, CAN_H : Pin 4, CAN_L : Pin 5
Female General Characteristics	Specification
Number of Sockets	5
Coupling Nut	Female
Coupling Nut Thread	M12 x 1 6G fit
Rotation <sup>1</sup>	Required
Standard	IEC 60947-5-2, Figure D.2 intermateability requirements
Pinout	Drain : Pin 1, V+ : Pin 2, V- : Pin 3, CAN_H : Pin 4, CAN_L : Pin 5
Physical Characteristics	Specification
Wiping Contact Plating Requirements	30 micro inch gold minimum over 50 micro inch nickel minimum or 5 micro inch gold minimum over 20 micro inch palladium-nickel minimum over 50 micro inch nickel. All gold must be 24 Karat
Wiping Contact Life	1000 insertion extractions.
Electrical Characteristics	Specification
Operating Voltage	25 volt minimum
Contact Rating	3 amps minimum
Contact Resistance	Nominal: Less than 1 mOhm Maximum: 5 mOhms over life
Environmental Characteristics	Specification
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL-1277, OIL RES II
Operating ambient temperature	-40 to +70 deg C full power with linear derating to 0 amps at 80 deg C.
Storage temperature	-40 to +85 deg C
<sup>1</sup> This requirement only applies to connectors on cable assemblies/cordsets	

Figure 8-3.42 Micro Connector Pinout

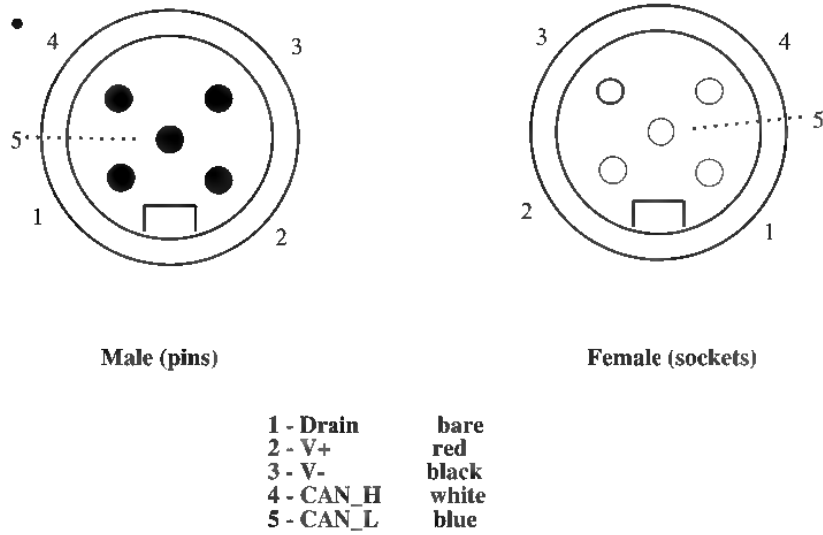


Figure 8-3.43 Micro 90 Degree Connector



### 8-3.12.5 Flat Trunk Connector Profile

**Table 8-3.78 Flat Trunk Connector**

Male General Characteristics	Specification
Number of Pins	4
Coupling Nut	(Optional retention with #8 x 1 ¾ screw)
Coupling Nut Thread	(UNC 32)
Rotation	None
Standard	See Figure 8-3 45
Pinout	CAN_L : Pin 1, CAN_H : Pin 2, V+ : Pin 3, V- : Pin 4
Female General Characteristics	Specification
Number of Sockets	4
Coupling Nut	(Optional retention with #8 insert or nut)
Coupling Nut Thread	(UNC 32)
Rotation	None
Standard	See Figure 8-3.45
Pinout	CAN_L : Pin 1, CAN_H : Pin 2, V+ : Pin 3, V- : Pin 4
Physical Characteristics	Specification
Wiping Contact Plating Requirements	8 micro inch gold minimum flash over 50 micro inch nickel. All gold must be 24 Karat
Wiping Contact Life	100 insertion-extractions.
Electrical Characteristics	Specification
Operating Voltage	25 volt minimum
Contact Rating	3 amps minimum
Contact Resistance	Nominal: Less then 1 mOhm Maximum: 5 mOhms over life.
Capacitance Between V+ and V-	0.22 uF +/- 10%
Environmental Characteristics	Specification
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL-1277, OIL RES II
Operating ambient temperature	-40 to +70 deg C full power with linear derating to 0 amps at 80 deg C.
Storage temperature	-40 to +85 deg C

The example in Figure 8-3.44 provides an illustration of a Flat Trunk Connector interface. The Flat Trunk Connector Profiles defines the mating interface between top and bottom modules shown in this example. For instance, if a Device would like to connect directly to the flat media without a drop cable, they may use this interface to connect to an existing two-piece connector that connects to the flat cable. This interface is not required when using a single piece design that directly connects to the flat cable.

**Figure 8-3.44 Example of Flat Trunk Connector**

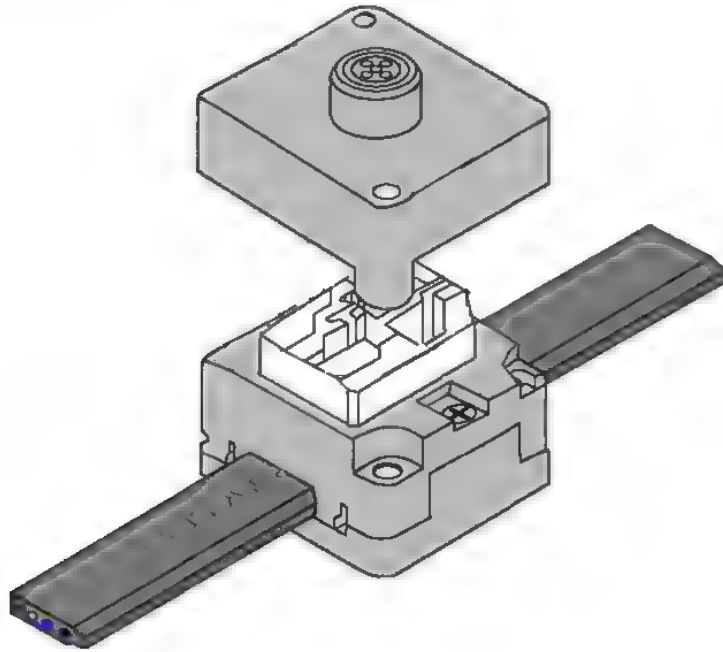


Figure 8-3.45 Flat Trunk Connector Layout

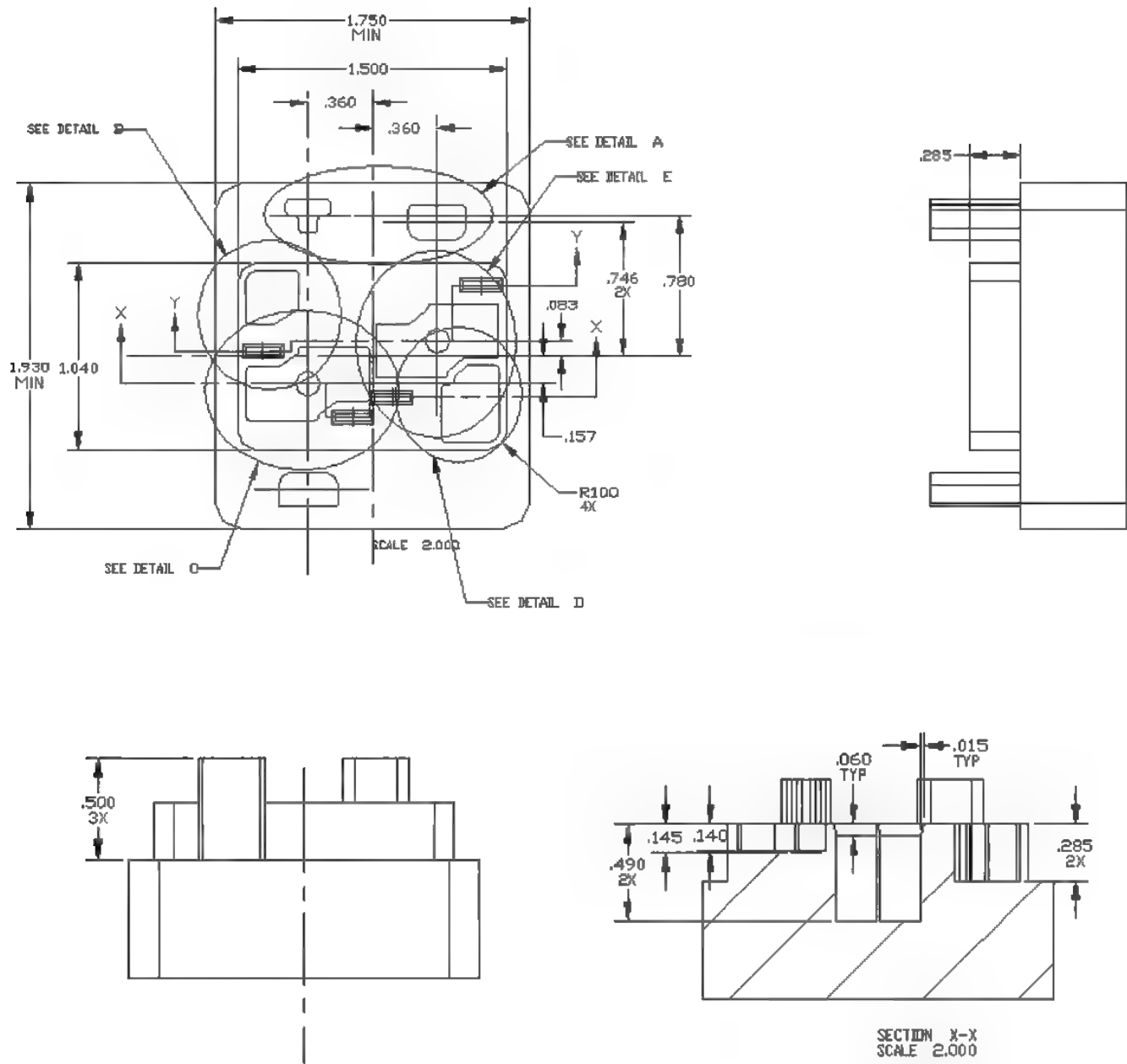
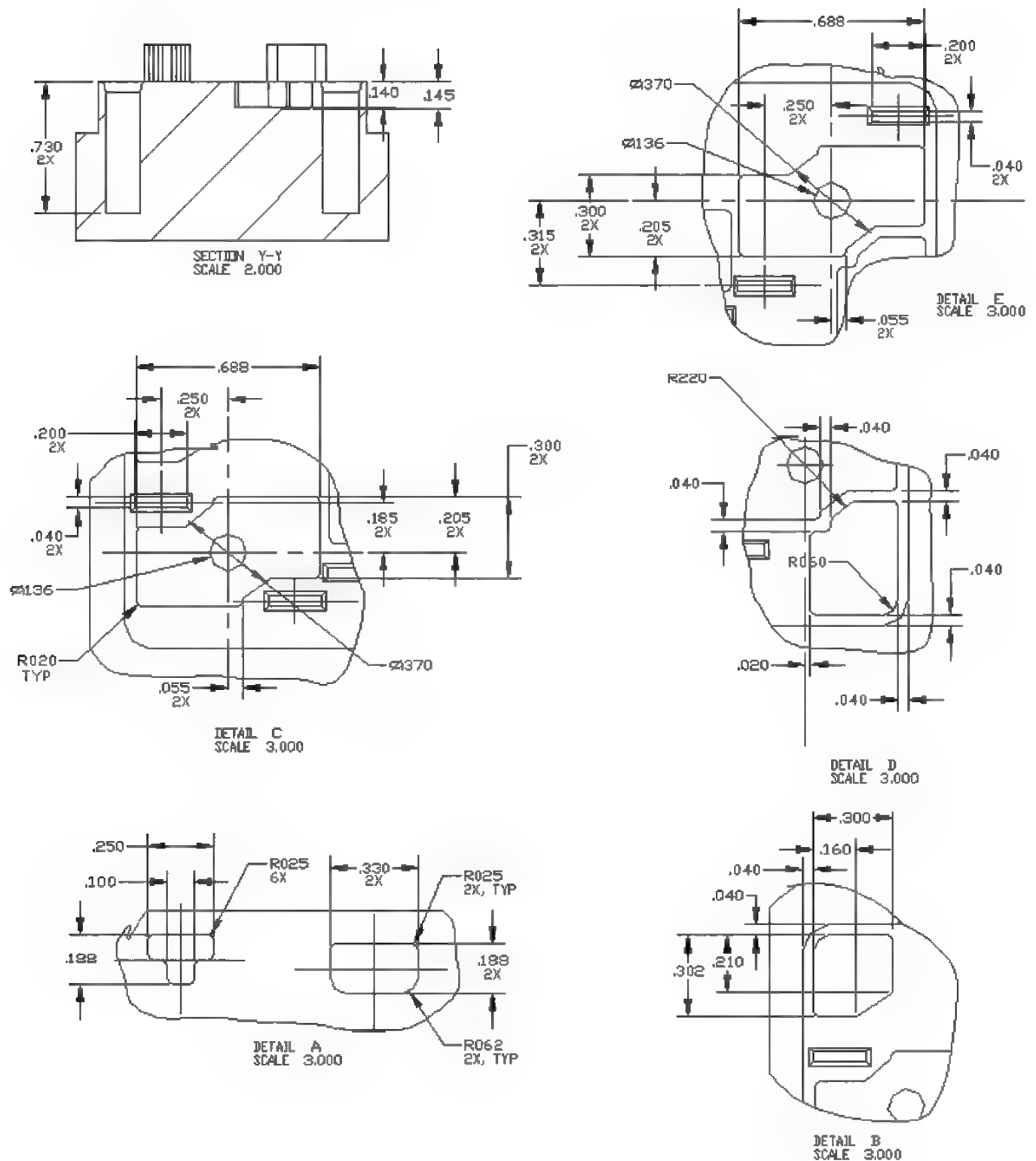


Figure 8-3.46 Flat Trunk Connector Layout (cont.)



### 8-3.12.6 Flat II Trunk Connector Profile

**Table 8-3.79 Flat II Trunk Connector**

Male General Characteristics	Specification
Number of Pins	4
Coupling Nut	N/A
Coupling Nut Thread	N/A
Rotation	N/A
Standard	See Figure 8-3.51
Pinout	V+: Pin1, CAN_H: Pin2, CAN_L: Pin3, V-: Pin4
Female General Characteristics	Specification
Number of Pins	4
Coupling Nut	N/A
Coupling Nut Thread	N/A
Rotation	N/A
Standard	See Figure 8-3.47 and Figure 8-3.48.
Pinout	V+: Pin1, CAN_H: Pin2, CAN_L: Pin3, V-: Pin4 with an internal 0.1uF/50V capacitor between V+ and V-
Physical Characteristics	Specification
Wiping Contact	Nickel under plated over 1.5µm,
Plating Requirements	Gold plated over 0.4µm
Wiping Contact Life	100 insertion-extractions
Contact physical dimensions	Conform to Figure 8-3.52 and Figure 8-3.53.
Electrical Characteristics	Specification
Operating Voltage	30V minimum
Contact Rating	5 A(Initial) 50mΩ maximum(Final) IEC60512-1 See Figure 8-3.51
Contact resistance	40mΩ maximum

Environmental Characteristics	Specification
Water resistance	N/A
Oil resistance	N/A
Operating ambient temperature	-35 to 70°C
Storage temperature	-35 to 80°C
Mechanical shock	Acceptance and rejection criteria. During the test, no current interruption shall occur 1µsec or more. Contact resistance / 50mΩ or less. Appearance and shape / No physical damage. Test Conditions. Acceleration: 490m/s <sup>2</sup> ; Duration: 11msec; 3 times in each of the 6 direction in 3 axes (Total: 18 times).
Vibration	Acceptance and rejection criteria: During the test, no current interruption shall occur 1µsec or more. Contact resistance / 50mΩ or less. Appearance and shape / No physical damage Test Conditions. Frequency: 10 to 500 Hz Sweep time: 20 min Amplitude: 1.52mm or 98m/s <sup>2</sup> 2 hours in each of the 3 axes (Total 6 hours).
Cables	Specification
Connectable cable	Flat II cable Female, Flat II and Thin Male

Figure 8-3.47 Flat II Trunk/Drop Connector (Male)

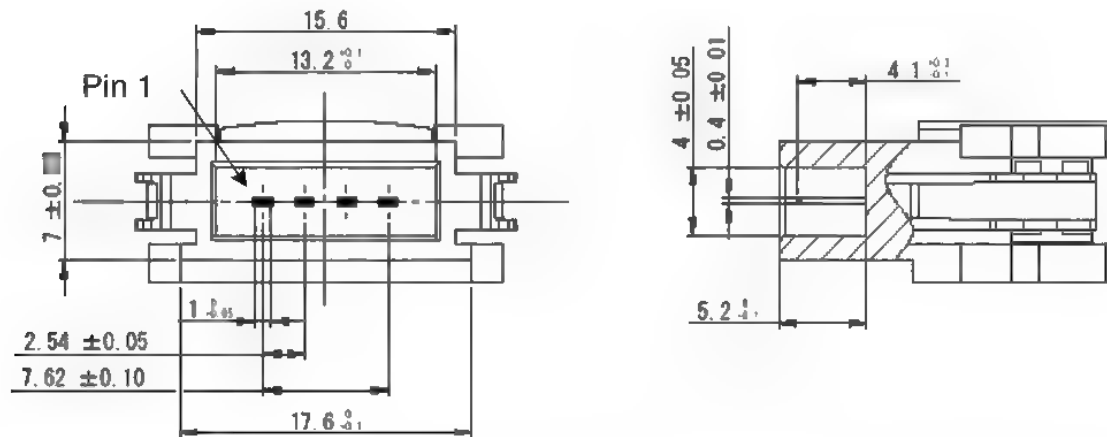




Figure 8-3.48 Flat II Trunk/Drop Connector (Male)

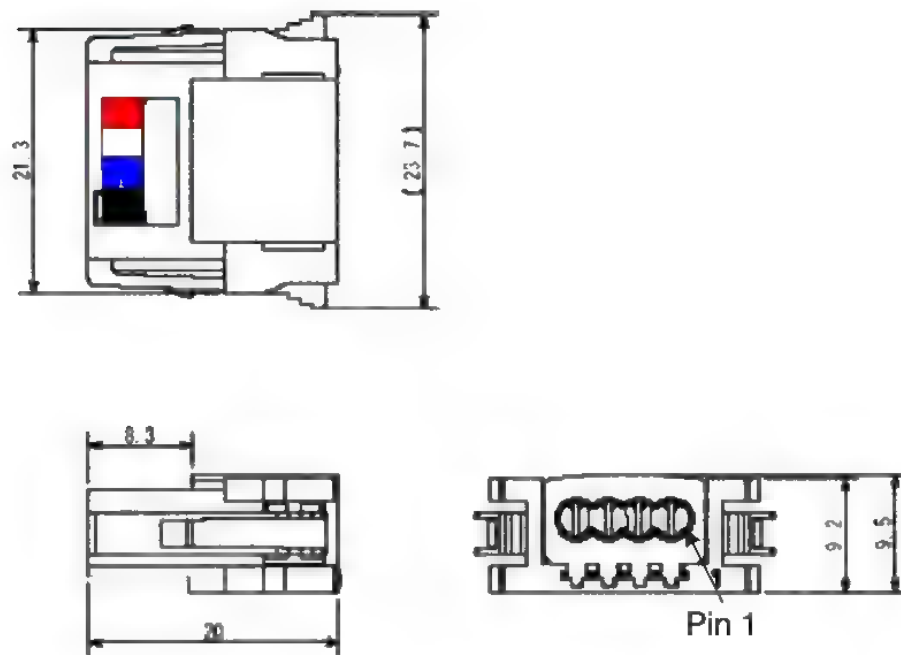


Figure 8-3.49 Flat II Trunk Connector (Female)

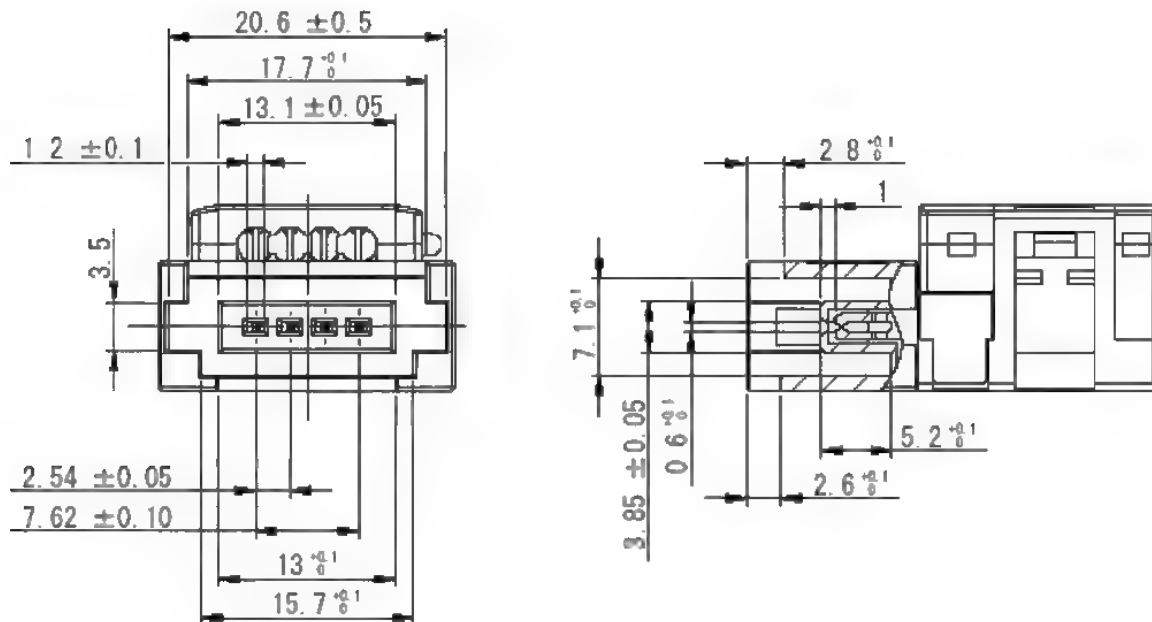


Figure 8-3.50 Flat II Trunk Connector (female)

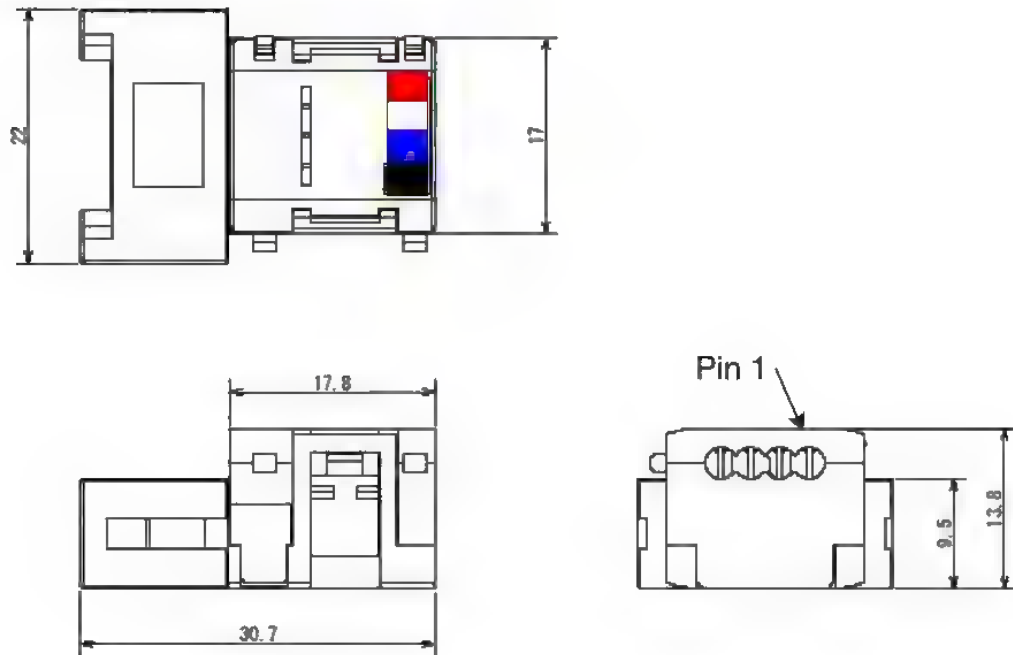


Figure 8-3.51 Measurement Method of Contact Resistance (Trunk Connectors mated)

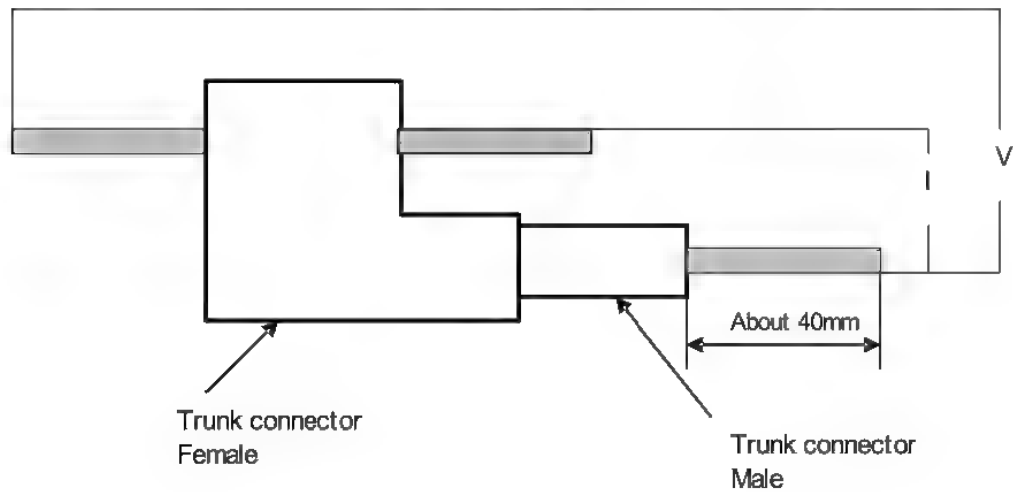


Figure 8-3.52 Flat II Socket Contact

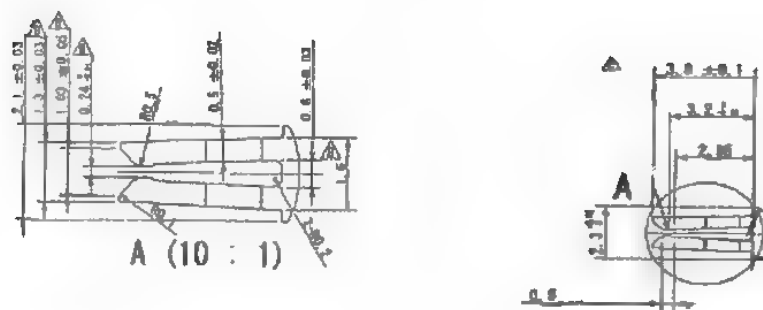
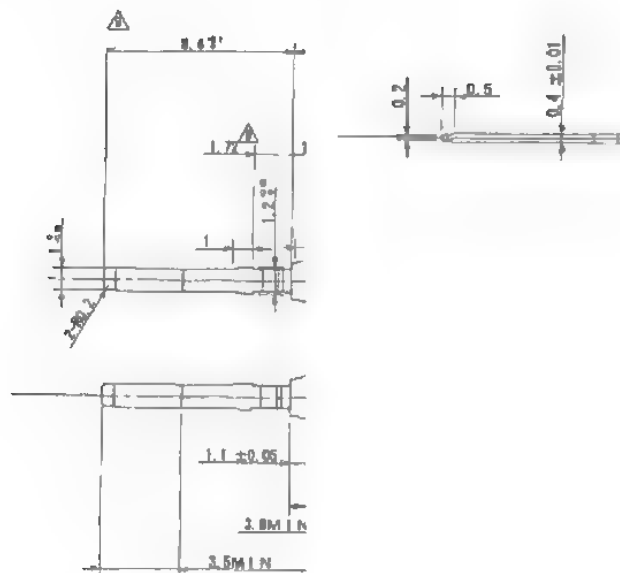


Figure 8-3.53 Flat II Pin Contact



### 8-3.12.7 Hard-wired Connections Profile

**Table 8-3.80 Hard-wired Connections**

Male General Characteristics	Specification
Number of pins (circuits <sup>1</sup> )	5
Coupling Nut	None
Coupling Nut Thread	None
Rotation	None
Standard	None
Pinout (Conductor order)	V+, CAN_H, Shield, CAN_L, V- (See 9-4.2.1)
Female General Characteristics	Specification
Number of sockets (circuits <sup>1</sup> )	5
Coupling Nut	None
Coupling Nut Thread	None
Rotation	None
Standard	None
Pinout (Conductor Order)	V+, CAN_H, Shield, CAN_L, V- (See 9-4.2.1)
Physical Characteristics	Specification
Terminal Plating (Wire Termination End)	100 micro inch tin minimum
Terminal Life	None
Electrical Characteristics	Specification
Operating Voltage	25 volt minimum
Terminal Rating	4 Amps Min.
Terminal Resistance	Nominal: Less than 1 mOhm Maximum: 5 mOhm over life
Environmental Characteristics	Specification
Water Resistance	None
Oil Resistance	None
Operating ambient temperature	-40 to +70 deg C full power with linear derating to 0 amps at 80 deg C.
Storage temperature	-40 to +85 deg C
1 Circuits relate to the number of connections made to a hard-wired connector, such as screw terminal blocks and barrier strips.	

Chapter 8-3.3 states that nodes shall be capable of being removed without severing the trunk.

Hard-wired connections, such as direct soldering, crimping, screw terminal blocks and barrier strips, are allowed.

**Important:** Whatever connection solution is chosen, it is mandatory that nodes shall be capable of being removed without severing or disturbing the network.

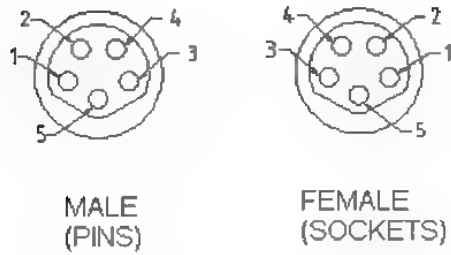
**Important:** Devices using hard-wired connections shall have clearly marked terminals and have the same order of wiring connection as defined in 9-4.2.1.

### 8-3.12.8 Sealed M8 Connector Profile

**Table 8-3.81 Sealed M8 Connector Profile**

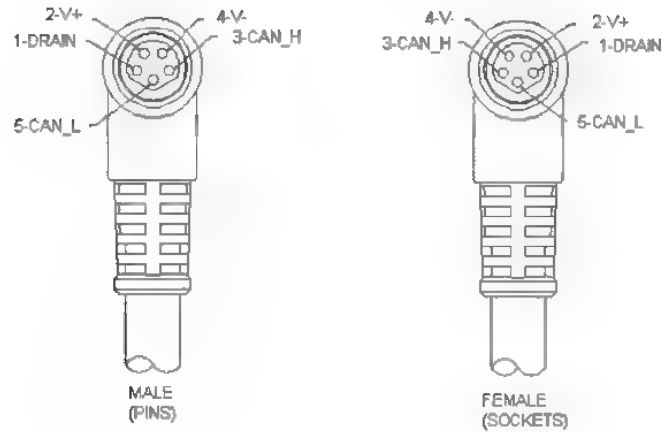
Male General Characteristics	Specification
Number of Pins	5
Coupling Nut	Male
Coupling Nut Thread	M8 x 1 6G fit
Rotation <sup>1</sup>	Optional
Standard	IEC 61076-2-104 Edit.on 1 Figure 18 and 19 B-Coding views
Pinout	Drain: Pin1, V+: Pin2, CAN_H: Pin3, V-: Pin4, CAN_L: Pin 5
Female General Characteristics	Specification
Number of Pins	5
Coupling Nut	Female
Coupling Nut Thread	M8 x 1 6G fit
Rotation <sup>1</sup>	Required
Standard	IEC 61076-2-104 Edit.on 1 Figure 18 and 19 B-Coding views
Pinout	Drain: Pin1, V+: Pin2, CAN_H: Pin3, V-: Pin4, CAN_L: Pin 5
Physical Characteristics	Specification
Wiping Contact Plating Requirements	30 micro inch gold minimum over 50 micro inch nickel minimum or 5 micro inch gold minimum over 20 micro inch palladium-nickel minimum over 50 micro inch nickel. All gold must be 24 Karat.
Wiping Contact Life	1000 insertion-extractions
Electrical Characteristics	Specification
Operating Voltage	25 volt minimum
Contact Rating	3 amps minimum
Contact resistance	Initial maximum: 5 mOhm Maximum change: 5 mOhms over life
Environmental Characteristics	Specification
Water resistance	IP67 and NEMA 4, 6, 6P, 13
Oil resistance	UL-1277 OIL RES II
Operating ambient temperature	-40 to +70°C full power with linear derating to 0 amps @ 80°C
Storage temperature	-40 to +85°C
<sup>1</sup> This requirement only applies to connectors on cable assemblies/cordsets	

Figure 8-3.54 M8 Connector Pinout



1-Drain	bare
2- V+	red
3- CAN H	white
4-V-	black
5-CAN L	blue

Figure 8-3.55 M8 90 Degree Connector



### 8-3.13 Transceiver

A transceiver is the physical component that provides transmission and reception of CAN signals onto and off of the network. The transceiver differentially receives signals from the network to provide to the CAN controller, and differentially drives the network with signals from the CAN controller.

**Important:** Several integrated CAN transceivers are commercially available. In selecting a transceiver, be sure to select one, which is capable of meeting the DeviceNet specifications. The remainder of this section contains detailed Physical Layer specifications and schematic designs.

**Important:** To be compatible with the power system design, the transceiver must support a minimum of  $\pm 5V$  common mode operation, which implies ground differences of  $\pm 5V$ .

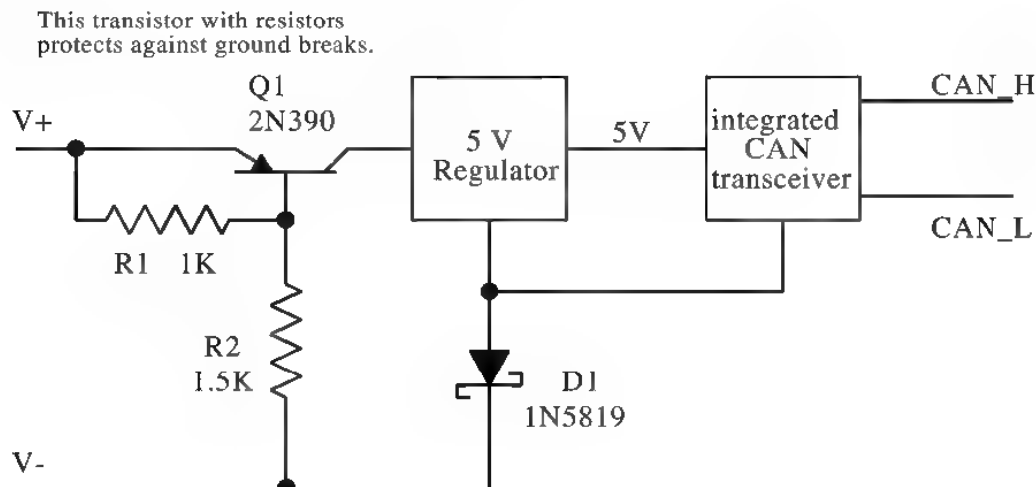
**Important:** An un-powered transceiver may have a lower input impedance than one that is powered. This causes unnecessary network loading and signal attenuation. A powered or un-powered physical layer must meet the differential input impedance as stated in Table 8-2.1, General Physical Layer Requirements.

#### 8-3.13.1 Mis-wiring Protection

DeviceNet requires that the node be capable of sustaining mis-connection of any of the wires on the connector in any combination. No permanent damage may occur in this situation with voltages as defined in section 8-2.1, including V+ voltages up to 18 volts.

Many of the integrated CAN transceivers have limited maximum negative voltage specifications on the CAN\_H and CAN\_L pins. To use these components, it is necessary to provide external protection circuitry. Figure 8-3.56 diagrams an example of this circuitry. A Schottky diode is inserted in the ground path to prevent accidental connection of the V+ signal to the V- terminal. A transistor switch inserted into the supply path prevents damage that can occur due to a loss of V- connection.

**Figure 8-3.56 Mis-Wiring Protection Example Circuit**



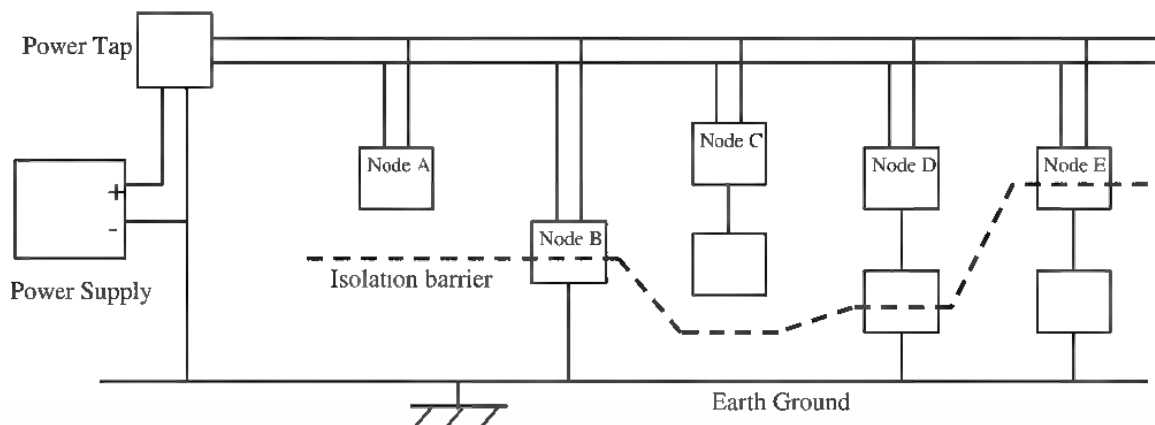
The values for Q1, R1, and R2 in Figure 8-3.56 are for reference only. These values must be appropriate for the application as determined by the developer. Q1 must be capable of the maximum expected current. R2 should be selected to provide adequate base current (usually  $I_c/20$  to  $I_c/10$ ) at minimum  $V_+$  (11 volts). If the dissipation/size of R2 becomes undesirable, and the regulator can handle lower input voltages, a Darlington transistor may be more desirable. R1 should be selected to draw several hundred micro amps but not more than several milli-amps. The base resistor limits the breakdown current should  $V_+$  and  $V_-$  be reversed. A diode may be added in the emitter, base, or across the base-emitter to eliminate avalanche if desired.

### 8-3.13.2 Grounding and Isolation

To prevent ground loops, the DeviceNet network should be earth grounded in only one location. The physical layer circuitry in all devices is referenced to the  $V_-$  bus signal. Connection to earth ground is provided at the bus power supply and is discussed in section 8-5. No significant current flow between  $V_-$  and earth ground may occur via any device other than a power supply under normal operating conditions. Allowable current is 1mA as defined in section 8-2.1. A ground isolation barrier must exist for every device as shown in Figure 8-3.57. Note that this barrier may exist external to the DeviceNet product. For example, in the case of some proximity sensors, the barrier may be plastic mounting hardware. Some products may have more than one potential path to ground. In these products, isolation must exist for all possible paths.

This isolation barrier is to accommodate the power supply voltage of 24VDC plus additional voltage drop, which could develop across the barrier due to ground potential differences. Products shall pass testing at  $\pm 500$ VDC as defined in section 8-2.1. Higher working voltages or test voltages may be supported based upon specific application requirements. In these cases, higher rated voltages and any additional tests will be specified by the vendor.

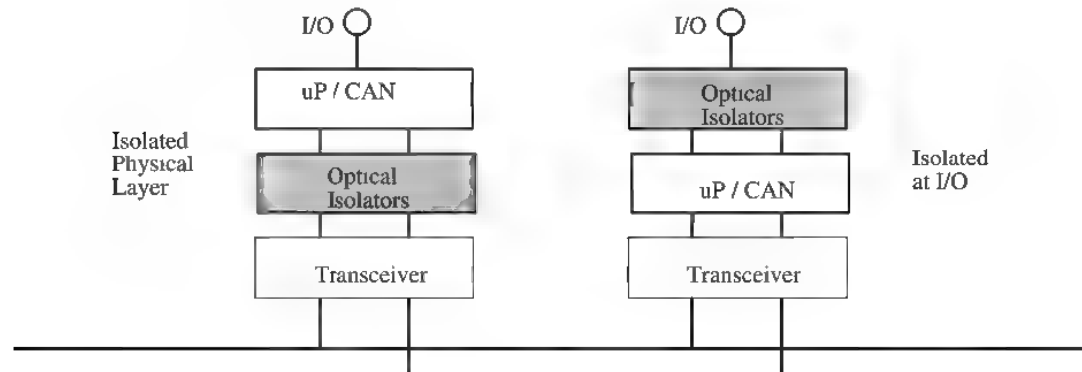
Figure 8-3.57 Ground Isolation on DeviceNet





Nodes B and E in Figure 8-3.57 contain the ground isolation barrier. A node which contains ground isolation is called an isolated node. A node which does not contain ground isolation is called a non-isolated node or an un-isolated node. An isolated node may either be isolated within the Physical Layer, or at some other location. Figure 8-3.58 shows two diagrams of isolated nodes. One contains isolation in the Physical Layer, and the other contains isolation at the I/O.

Figure 8-3.58 Two types of isolated nodes



### 8-3.14 Non-isolated Physical Layer

DeviceNet supports both non-isolated and isolated Physical Layers within nodes.

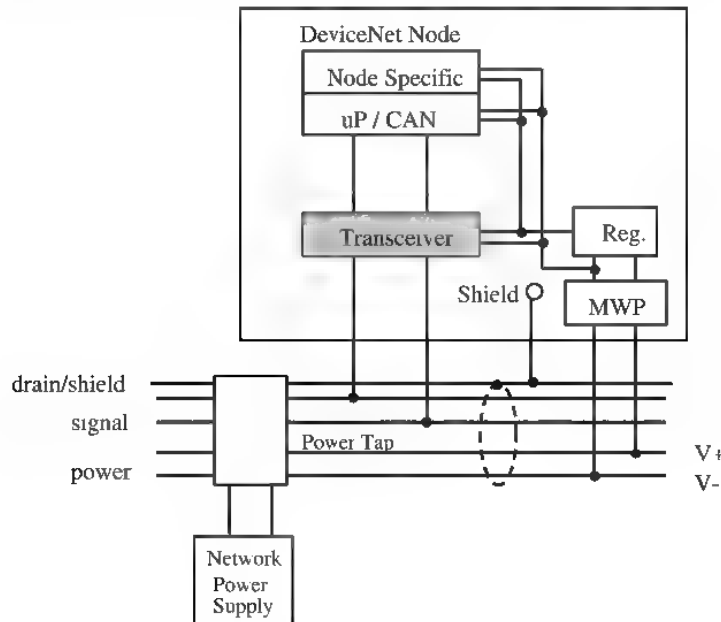
**Important:** If a node contains a *non-isolated physical layer*, all components of that node must be either ground referenced to V- or elsewhere ground isolated.

**Important:** Within a node that contains a non-isolated physical layer, components ground referenced to V- may connect to other external devices through serial ports, parallel ports, or I/O connections as depicted by Node C and Node D in Figure 8-3.57. These external devices must be ground isolated and this requirement must be stated in the product literature of any DeviceNet product allowing this type of connection.

The drain connection on the DeviceNet connector should be connected through a parallel R-C to the device enclosure. See the example schematics in section 8-2.2. For best EMI performance, the conductors should be kept very short along this path, and the enclosure should be a closed structure of conductive material. Should the device not have such an enclosure, this pin on the connector may be left unconnected.

Figure 8-3.59 shows an example of a node with a non-isolated physical layer, which obtains all of its power from the network and has all of its components ground referenced to V-. Drain wire termination detail is not shown.

Figure 8-3.59 Node containing a Non-isolated Physical Layer



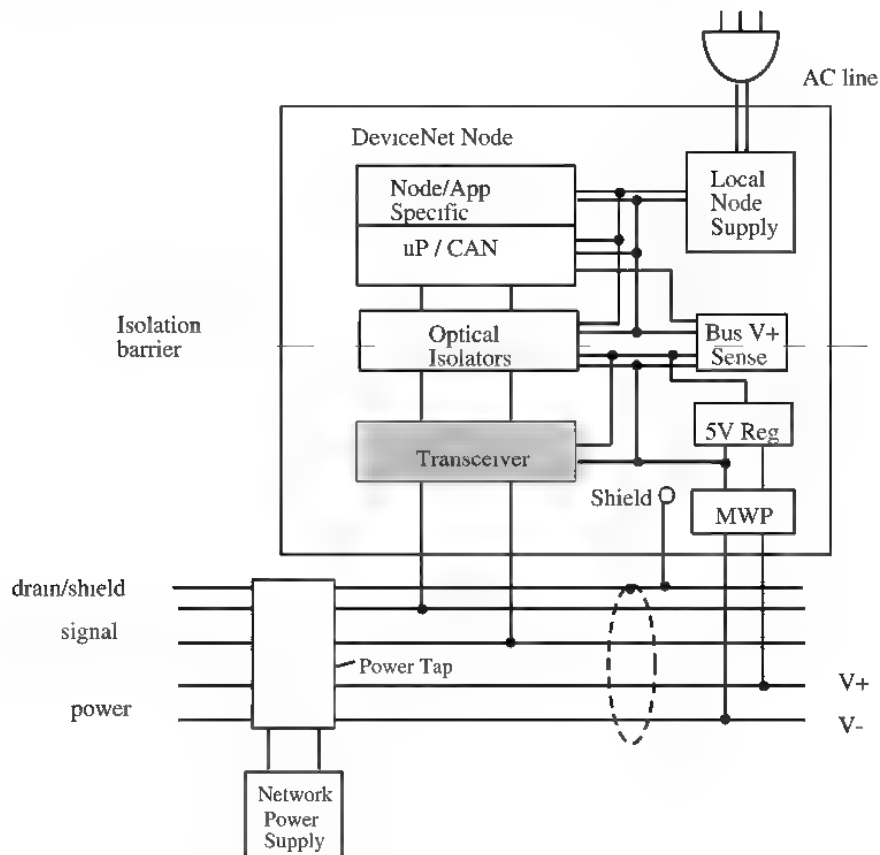
### 8-3.15 Isolated Physical Layer

Within a node that contains an *isolated physical layer*, *some* of its components are referenced to a ground other than V- of the DeviceNet network. This includes such things as devices with serial ports, parallel ports, RS232 and RS485 type ports, or anything that has a ground other than V- of the network.

Nodes containing an isolated physical layer, such as the one in Figure 8-3.60, must meet the differential input impedance as stated in Table 8-2.1 regardless if in a powered or un-powered state. Network power may be used for other circuitry provided that the network powered circuitry is ground referenced to V-, or is otherwise ground isolated. Drain wire termination detail is not shown in this figure.

The drain connection on the DeviceNet connector should be connected through a parallel R-C to the device enclosure. See the example schematics in section 8-2.2. For best EMI performance, the conductors should be kept very short along this path, and the enclosure should be a closed structure of conductive material. Should the device not have such an enclosure, this pin on the connector may be left unconnected.

**Figure 8-3.60 Node containing an Isolated Physical Layer**



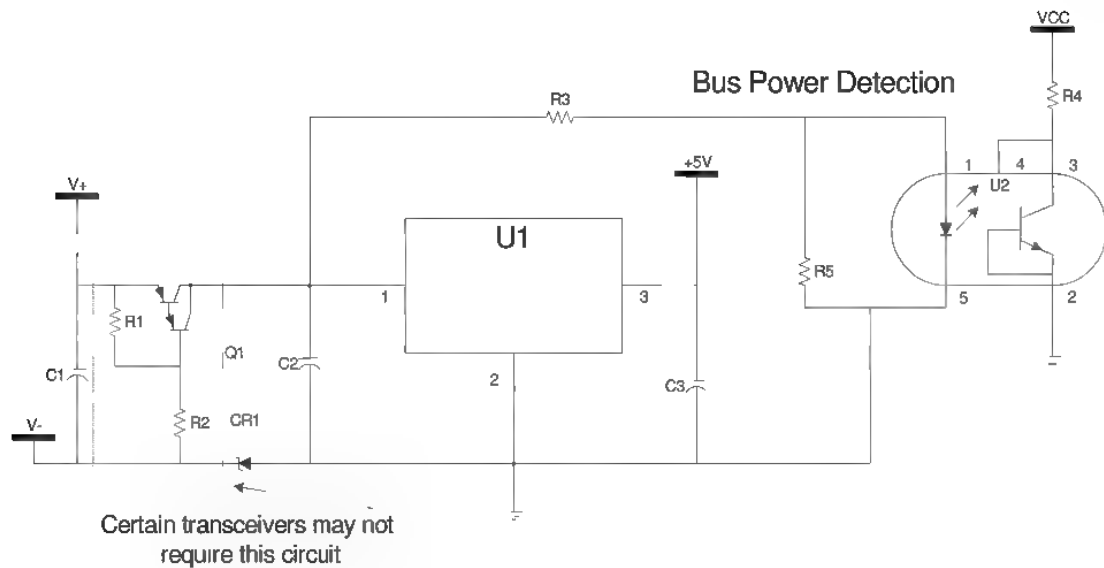
**Important:** If you use opto isolators, high-speed capability is necessary. The critical parameter is propagation delay. To meet the DeviceNet specification, a maximum delay of 40ns is appropriate. The combined delays of the opto isolators and the transceiver must meet the requirements in section 8-2.1.

Note that isolated nodes and non-isolated nodes can exist simultaneously and communicate on DeviceNet. The opto delays are taken into account in the maximum trunk length limits.

**Important:** For an isolated node, with physical layer isolation, the CAN controller is still active during a de-energized bus condition. A node not having knowledge of this event will eventually enter a bus-off state requiring a reset. A node must not enter a bus off condition whereby requiring user intervention in the event of a transceiver power failure or a network power failure. An example circuit for accomplishing this task is shown in Figure 8-3.61.

**Important:** When bus power is de-energized and re-energized, a node that is auxiliary powered may not have knowledge of this event. A node shall perform a duplicate MAC ID check when bus power is re-energized.

### Figure 8-3.61 Example Circuit for Bus Power Detection



## 8-4 Powered Node Implementation

DeviceNet offers users the ability to draw power from the network. This section explains how you can implement your devices using this power. Included are the following:

- Node power regulator
- Use of regulator in power design

### 8-4.1 Node Power Regulator

A power regulator located on each device takes power from DeviceNet and converts it to the level required locally. For example, if a device requires 5 volts, then the regulator will reduce the network voltage from 11 to 25 V down to 5V.

Power converters for DeviceNet must have the following characteristics:

- Input voltage range of 11-25 volts
- Node filtering (to keep noise off the +24 VDC bus)
- Transient protection
- Either a linear or switching regulator
- Delayed start (to reduce turn-on surge) for switching regulators

**Important:** Each node must have either a linear or switching regulator to reduce the nominal 24V from the bus to 5V locally. The current required determines the type of regulator you should use. Refer to Table 8-4.1 to determine the appropriate regulator for your application. Linear regulators are recommended for currents up to 50 mA but not above this due to their inefficient use of power.

**Table 8-4.1 Recommended Regulator Type Based on Bus Current**

If current is:	Recommended Solutions:
Less than 50 mA	Linear regulator
Between 20 mA and 5 amps	Switching regulator

## **8-4.2 Use of Regulator in Power Design**

A node can get power from the DeviceNet bus, from an outside source, or from a combination of both. However, regardless of the source, a regulator is required at each device.

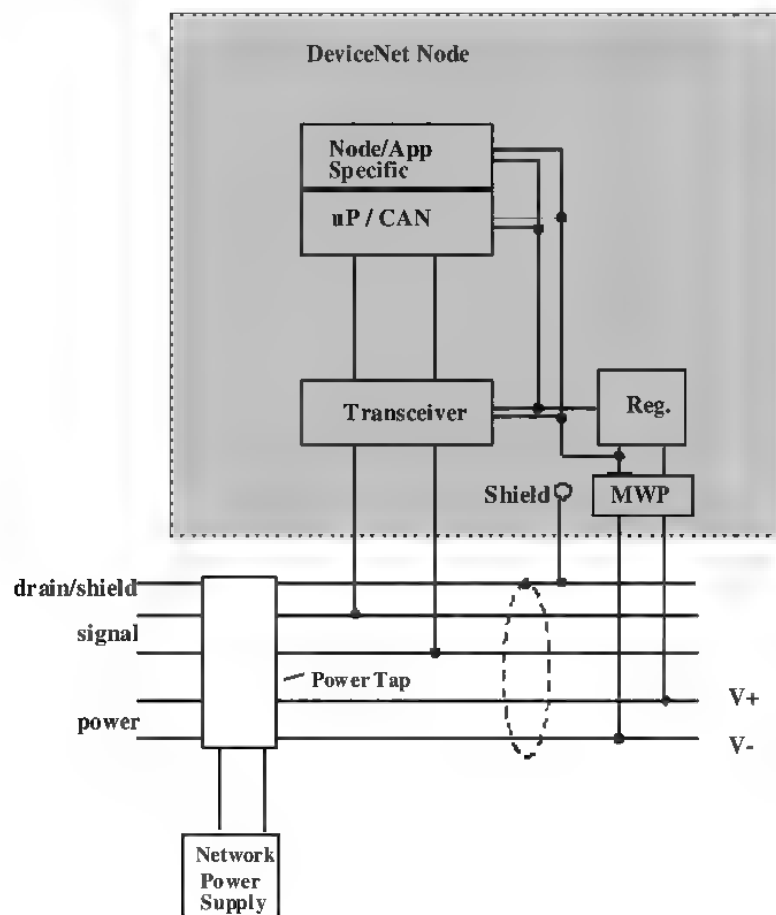
While variations exist, four basic power implementations are most common. Notice that all require a voltage regulator. Figure 8-4.1 through Figure 8-4.4 illustrate these options:

- Non-isolated node, powered by the network
- Isolated node, powered by the network
- Isolated node, with transceiver powered by the network
- Isolated node, providing power to the network

### **8-4.2.1 Non-isolated Node Powered by the Network**

The network can power both non-isolated and isolated nodes. Figure 8-4.1 shows a non-isolated DeviceNet node powered solely by the network. Every component within the device depends on the network for power.

### Figure 8-4.1 Non-isolated Node Powered by the Network



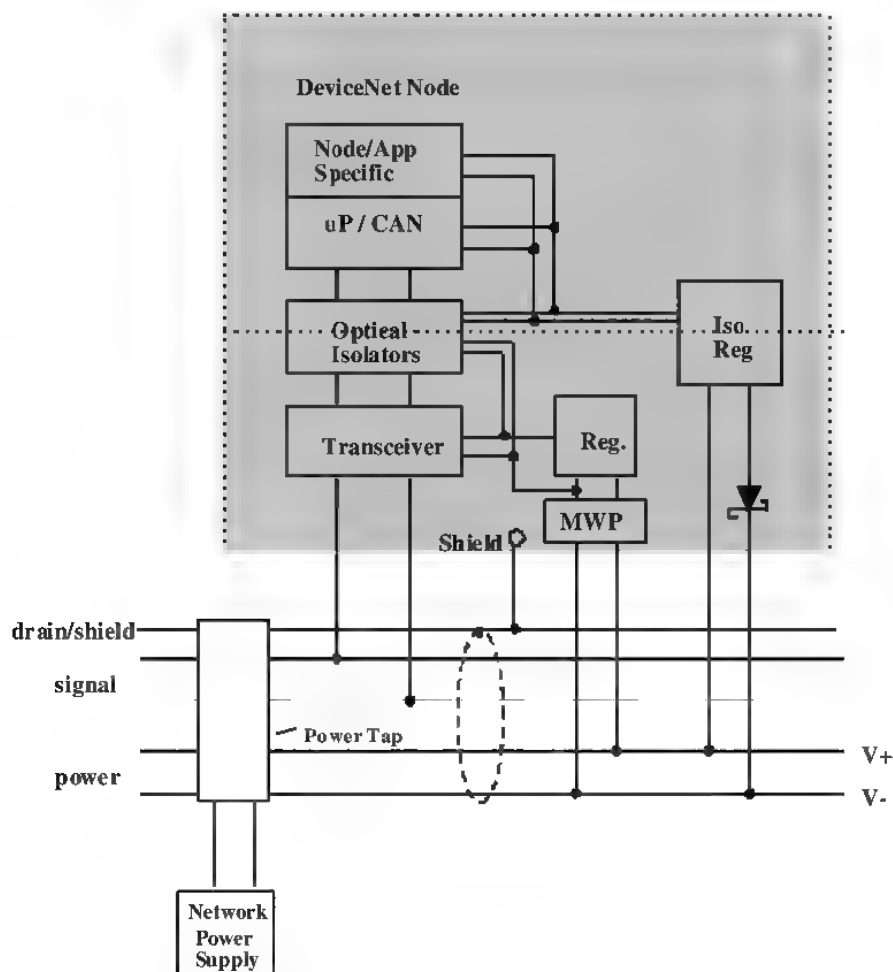
### 8-4.2.2 Isolated Node Powered by the Bus

Figure 8-4.2 shows an isolated node that gets its power from the network. The node is labeled “isolated” because not all of its components have the same ground reference. Yet, all components are still powered by the network.

The node in the figure contains two regulators: one is isolated and powers the CAN controller, node-specific application, and half the optos; while the other is non-isolated. The non-isolated regulator supplies the transceiver and the network half of the optos.

This approach is very flexible, yet complex and more expensive than the other methods.

Figure 8-4.2 Isolated Node Powered by the Network\*



\*Although this node is fully powered by the network, other non-isolated connections may require isolation.

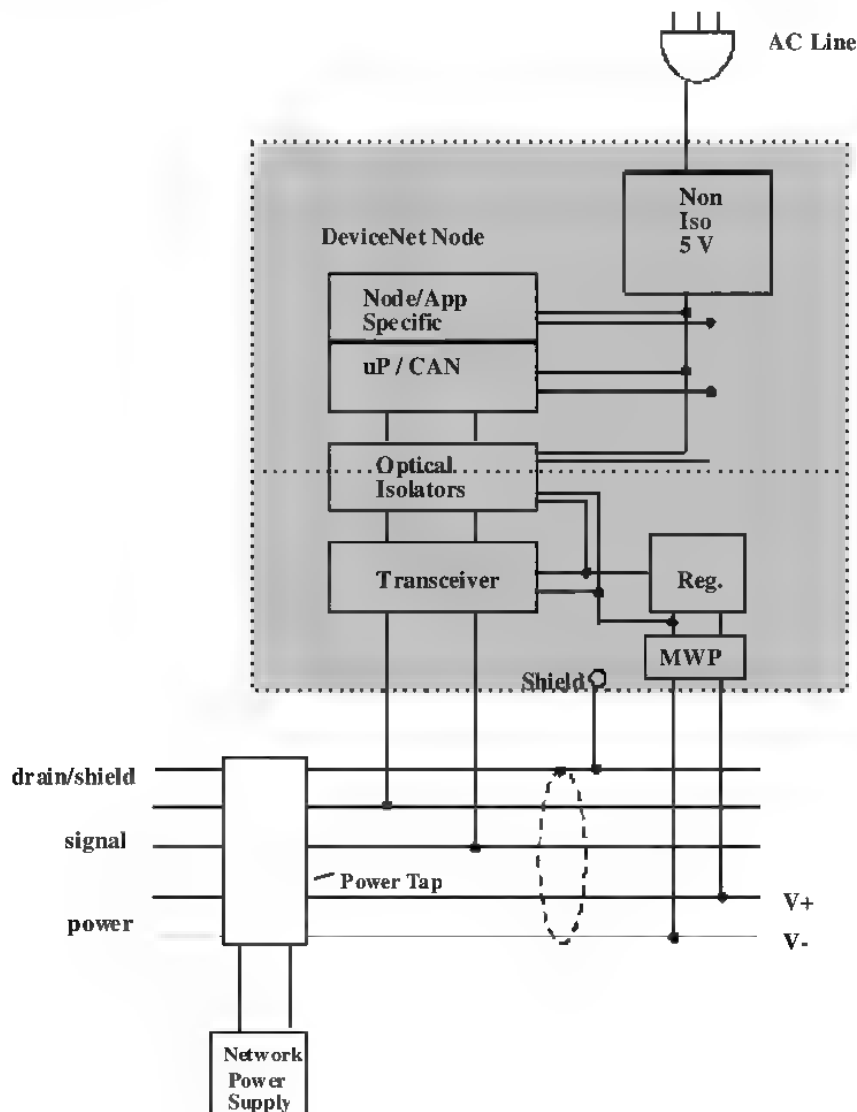
### 8-4.2.3 Isolated Node with Transceiver Powered by the Network

Figure 8-4.3 shows a node powered by both the network and another source. In this case, the transceiver and half of the optos are powered by the network. But the rest of the node is powered by the AC line.

This method is beneficial when an application requires a significant amount of power, and it is desirable not to heavily load the network.

Important: When bus power is de-energized and re-energized, a node that is auxiliary powered may not have knowledge of this event. A node shall perform a duplicate MAC ID check when bus power is re-energized.

Figure 8-4.3 Isolated Node with Transceiver Powered by the Network



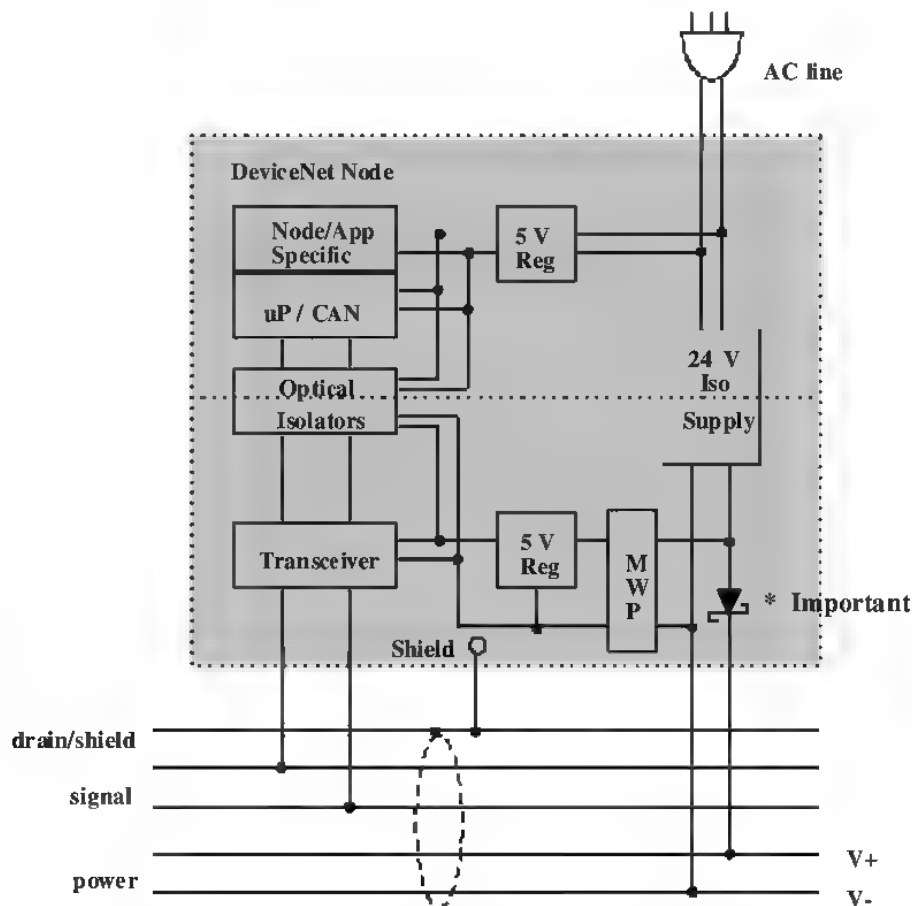


#### 8-4.2.4 Isolated Node Providing Power to the Network

Use this method when you have a limited number of devices on the network that don't require much power. The node can power the network and eliminate the need for separate power supplies.

In Figure 8-4.4, the AC line powers a regulator which provides five (5) volts locally. The AC line also powers a 24 volt isolated supply, which powers the network, and another five-volt regulator, which in turn, powers the transceiver and optos.

Figure 8-4.4 Isolated Node Providing Power to the Network



**Important:** \*If a node is going to supply power to the network without the use of a power tap, then it should include a Schottky Diode and over-current protection usually present in the power tap. See Schottky Diode attached to V+ of the network in Figure 8-4.4. Also, if the node is connected to the bus with a drop line the maximum current that can be delivered must meet the drop line current limits in section 8-5.2.

#### 8-4.2.5 Auxiliary Power Ports

For auxiliary power connector styles and pin outs see Section 8-2 and 8-3 of Volume 1 of the CIP Networks Library.

## 8-5 Configuring Network Power

In addition to providing communications, DeviceNet also provides power. Because power and signal conductors both are contained in the cable, devices can draw power directly from the network without the need for separate power sources.

DeviceNet has a single supply current capability of up to 16 amps depending on the cable selected. The maximum current for any cable is listed under the appropriate cable specifications in section 8-3.8. Smaller gauge cables provide a cost-effective means of designing a highly functional, flexible, low current cable system.

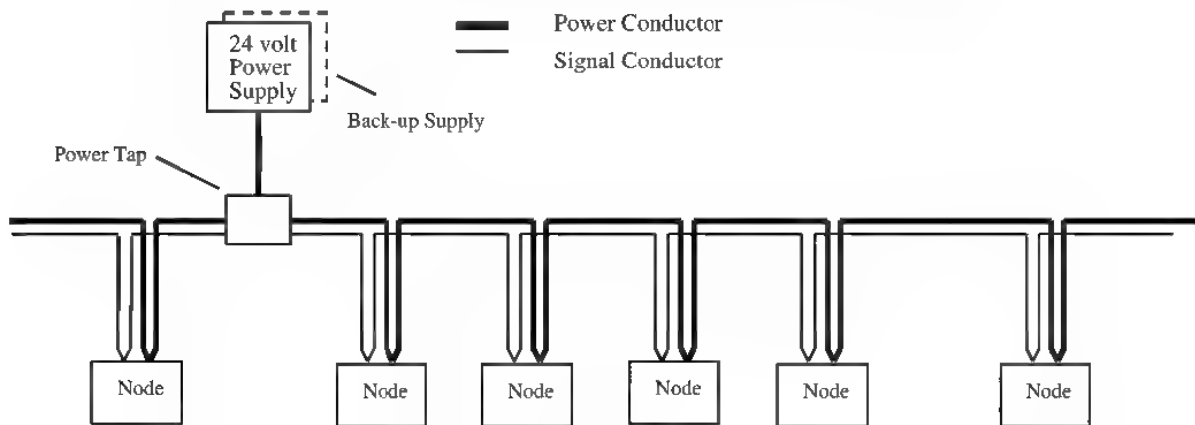
The power bus capabilities for DeviceNet are:

- Cable length as long as 500 m (1,640 feet)
- Support for as many as 64 nodes of varying current
- Adjustable configuration

Because of the flexibility of DeviceNet, there are several power design choices. This section provides guidelines to help configure power along a network in a way that maximizes performance and minimizes cost.

**Note:** For detailed specifications on the various aspects and components involved in power please see section 8-3.11.

**Figure 8-5.1 Power Along the Network with Both Signal and Power Conductors in Cable**



### 8-5.1 Defining Your Power Configuration

Power configuration is adjustable based on your system requirements.

The DeviceNet power bus is supplied by a nominal 24 volt source and can support up to 8 amps on any section depending on the type of trunk cable used (see section 8-3.8) or less current using small gauge cables. The maximum current capacity for each cable type is listed in section 8-3.8. Since this much current can be drawn from each side of a power tap, a single supply network can possibly provide twice these current levels. If the system has even greater requirements, DeviceNet can support multiple power supplies, which can result in almost unlimited power. The majority of DeviceNet applications, however, will require only one power supply.

**Important:** Before you begin, familiarize yourself with country and local codes in which the system to be installed. In the U.S. and Canada the DeviceNet some cable types must be installed as a Class 2 circuit when installed as building wiring. This requires limiting the current in any section to 4 amps. The rating of the system components themselves, however, is 8 amps.

The trunk line can be constructed of any of the single cable types within a segment. Mixing of cable some types in the same segment is permitted with the appropriate de-rating as defined in the individual cable profiles in section 8-3.8. There are several items that can limit the current available on the network. More than one of these limits can independently exist at the same time:

- Cable type (current capacity)
- Connector type (contact rating)

The maximum common mode limit of the transceiver\*

\*simply put, this parameter limits the voltage drop that can be tolerated on the power cable. The trunk cable current that corresponds to the maximum common mode voltage drop is indicated in section 8-3.8.

#### **8-5.1.1 Quick Start**

To begin configuring your power, start with the quick and easy steps below:

1. Add together the current requirements of each device on your network.
2. Measure the total length of your network.
3. Use section 8-3.8 to look up the current capability based on the length of your network and the type of cable being used.
4. If the accumulated current from step 1 is less than the value found in section 8-3.8, then any of the primary network configurations presented in the next section can be used.
5. If the accumulated current is greater then the value found in section 8-3.8, then continue to the next section. Note that section 8-5.1.2.1 duplicates the calculations of this section and can be skipped.

**Important:** The power supply capability must be equal to or greater than the load requirement on your network.

#### **8-5.1.2 Primary Network Configurations**

Depending on your power requirement and cable type used, use one of these configurations to minimize complexity and cost.

- Single supply end-connected
- Single supply center-connected

The following sections provide methods for determining the appropriate configuration for your network.

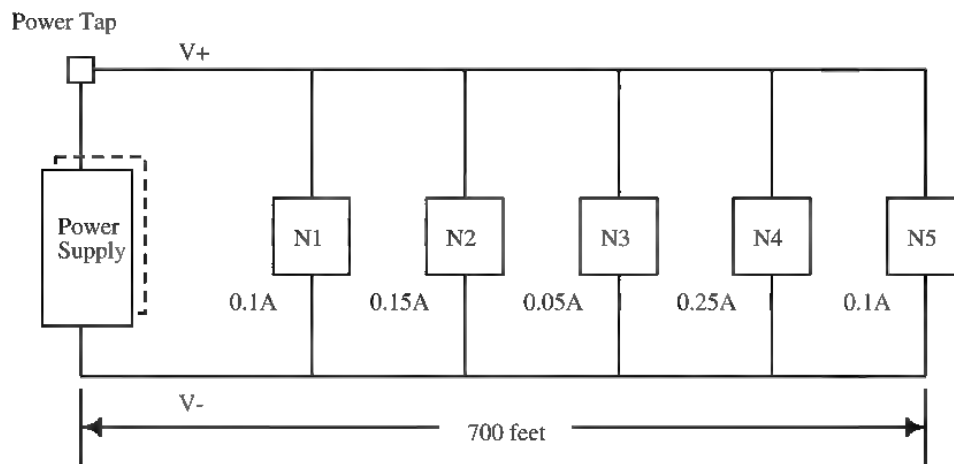
### 8-5.1.2.1 Single Supply End-connected

The calculation you performed in section 8-5.1.1, Quick Start, determined if your network supports this configuration. For example, using Thick cable, if the accumulated current requirement for your network exceeds the value found in section 8-3.8.2, Table 8-3.11, then this configuration is not appropriate for your network.

Single-supply, end-connected is the simplest configuration, but also provides the lowest power capability.

Use Figure 8-5.2 as an example to illustrate power configuration for a 700-foot network with a single supply end-connected using thick cable trunk line. The procedure would be the same as described above.

**Figure 8-5.2 Single Supply End-Connected**



#### **Calculation Method:**

Total network length = 700 feet

Total current =  $.1A + .15A + .05A + .25A + .1A = .65A$

Table 8-3.11 current limit for 700 feet = 1.5A

**Conclusion:** All configurations are acceptable for this network.

### 8-5.1.2.2 Single Supply Center-Connected

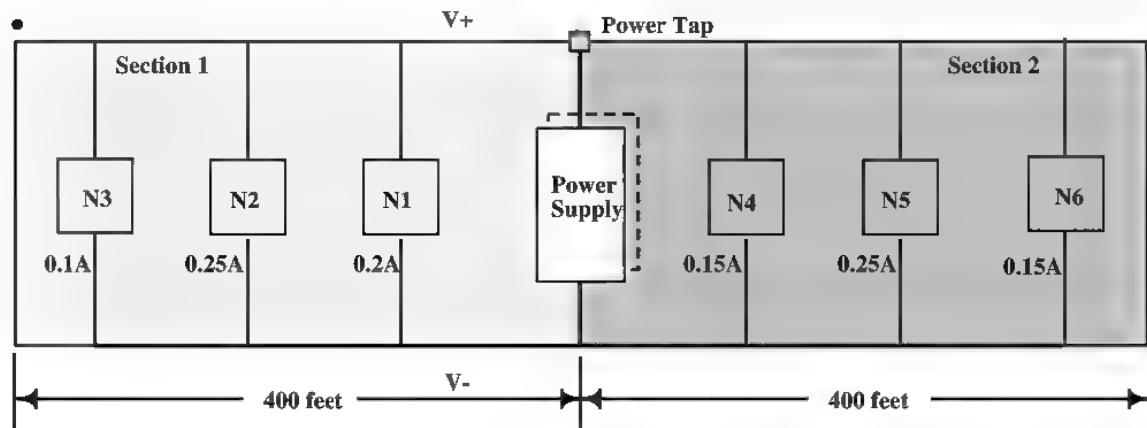
If previous calculations indicated that you must place your power supply somewhere other than at the end of the network, then use the following steps to determine the validity of a power supply connected at the Physical Center of the network.

A single center-connected supply provides *double* the total current capability of a single end-connected supply.

1. Add the current ratings of devices in each section to determine total current in each section. Because the supply is center-connected, the bus has two sections, one on each side of the power tap. See Figure 8-5.3.
2. Using Table 8-3.11 and the graph in Figure 8-3.10 for a thick cable trunk line, look up the maximum current capability for each section based on the length of each section, including the power supply drop cable. Use Table 8-3.16 and Figure 8-3.12 if thin cable trunk line is used on a section.

If the currents in both sections are less than the table value, then your network can support a single-supply, center-connected configuration. Use Figure 8-5.3 as an example to illustrate the appropriate calculations for thick cable trunk line.

**Figure 8-5.3 Single Power Supply Center-Connected**



**Calculation Method:**

Section 1 length = Section 2 length = 400 feet

Section 1 current =  $.1A + .25A + .2A = .55A$

Section 2 current =  $.15A + .25A + .15A = .55A$

Table 8-3.11 current limit for 400 feet (Section 1) = 2.63A

Table 8-3.11 current limit for 400 feet (Section 2) = 2.63A

**Conclusion:** Power supply located in center is feasible for both sections.

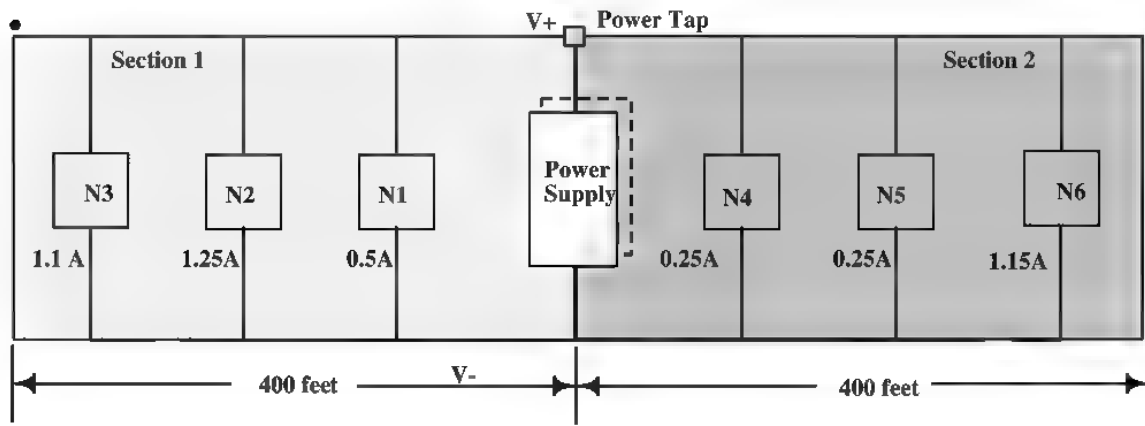
If your current (determined in Step 2.) in a given section exceeds the maximum current capability according to the appropriate table, then do one of the following, based on your circumstance:

**Table 8-5.1 Power Supply Recommendations**

If your current exceeds the maximum in:	then:
Only one of the two sections	Move the position of the supply along the overloaded section and recalculate; or Move the power supply so that one of the nodes from the overloaded section is now in the other section
Both sections	Use two power supplies

Figure 8-5.4 illustrates an example of a single-supply, center-connected network using thick cable trunk line with an overload:

**Figure 8-5.4 Single Power Supply Center-Connected with Overload**



**Calculation Method:**

Section 1 = Section 2 = 400 feet

Section 1 current =  $1.1A + 1.25A + .5A = 2.85A$

Section 2 current =  $.25A + .25A + .85A = 1.35A$

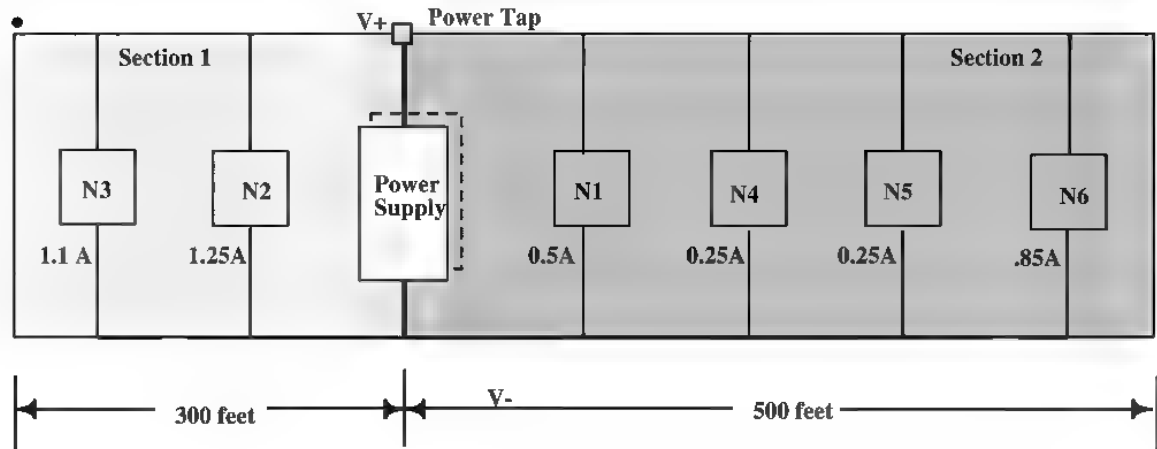
Table 8-3.11 current limit for 400 feet (Section 1) = 2.63A

Table 8-3.11 current limit for 400 feet (Section 2) = 2.63A

**Conclusion:** Overload present on section 1.

**Solution:** Move supply in direction of overloaded section, as shown in Figure 8-5.5.

Figure 8-5.5 Solution to Single Supply Center-Connected with Overload



**Calculation Method:**

Section 1 = 300 feet

Section 2 = 500 feet

Section 1 current =  $1.1\text{A} + 1.25\text{A} = 2.35\text{A}$

Section 2 current =  $0.5\text{A} + .25\text{A} + .25\text{A} + .85\text{A} = 1.85\text{A}$

Table 8-3.11 current limit for 300 feet (Section 1) = 3.51A

Table 8-3.11 current limit for 500 feet (Section 2) = 2.10A

**Conclusion:** Overload no longer present in either section.  
This configuration is now feasible.

## 8-5.2 Load Limits

The Maximum currents that can flow on the network power bus for each cable type can be found in section 8-3.8. The maximum drop line current depends upon the length of the drop line.

**Important:** The device operating current represents the average current drawn from the bus.

**Important:** If the maximum device operating current exceeds the average current by 10% then the maximum operating current must be specified in the product data sheet.

Current limits for drop lines are shown in Table 8-5.2 below and described by the following equations:

$$I = 15 / L$$

where I is the allowable drop line current in amps and L is the drop length in feet, or:

$$I = 4.57 / L$$

where I is the allowable drop line current in amps and L is the drop length in meters.

**Table 8-5.2 Maximum Allowable Drop Line Current**

Drop Length (ft)	Maximum Allowable Current (Amps)
1	3.0
3	3.0
5	3.0
7.5	2.0
10	1.5
15	1.0
20	0.75

## 8-5.3 System Tolerance

By using the stack up limits outlined in this table, it is possible to make performance trade-offs and still meet DeviceNet requirements. Maximum system voltage tolerance for DeviceNet is 24 volts +/- 4.0%. Table 8-5.3 shows the allocated tolerance budget.

**Table 8-5.3 Recommended Stack Up Standard for DeviceNet**

Specification	Parameter
Initial Power Supply Setting	1.0%
Line Regulation	0.3%
Load Regulation	0.3%
Temperature Drift	0.6% total
Schottky Diode Drop	0.75%
Time Drift	1.05%
Total Stack Up	4.0%



#### **8-5.4      Avoiding Errors**

We suggest taking the following steps to minimize the number of design and construction errors that could be encountered when configuring power on the network:

- Perform calculations suggested in section 8-5 to prevent the network from being overloaded.
- Conduct voltage measurements once the network is constructed to verify accurate configuration. A minimum of 11 volts at a node is required, and a maximum voltage drop (from V+ to V-) of 10 volts is permitted.
- Allow for sufficient margin of error to correct configuration problems without damaging the network.
- For heavily loaded multiple supplies: Turn on all supplies simultaneously to prevent overloading fuses or make sure that un-powered sections are disconnected from the rest of the network.

#### **8-5.5      Power Supply Options**

With the presence of a DeviceNet power tap requirement to protect the network from unlimited current flow, almost any off the shelf power supply can be used as long as they meet the general requirements listed below and the more detailed requirements of section 8-3.11.1.

- +24 VDC
- Ability to support linear and switching regulators
- Tolerance of: +24 VDC +/- 1% and current capability of 0-16 amps (single and multiple supply applications)
- Supply outputs must be isolated from the AC line and chassis

## **Volume 3: DeviceNet Adaptation of CIP**

# **Chapter 9: Indicators & Middle Layers**

## Contents

9-1	Introduction .....	3
9-2	Indicators ..	3
9-2.1	DeviceNet Indicator Requirements .....	3
9-2.2	Module Status LED.....	4
9-2.3	Network Status LED .....	6
9-2.4	Module and Network Status LEDs at Power-Up. ....	7
9-2.5	Combined Module/Network Status LED .....	8
9-2.6	Combined Module/Network Status LED at Power-Up .....	9
9-2.7	I/O Status LED.....	9
9-2.8	LED Flash Rate .....	10
9-3	Switches .....	10
9-3.1	DeviceNet MAC ID Switches .....	10
9-3.2	DeviceNet Baud Rate Switches .....	10
9-4	Recommended Physical Standards .....	11
9-4.1	Recommended Indicator and Switch Labeling .....	11
9-4.2	Recommended Connector Labeling .....	11
9-4.2.1	Open Style Connector (5-Position). ....	11
9-4.3	Recommended Connector Key Orientation .....	12
9-5	DeviceNet and CAN .....	13
9-6	The Scope of CAN... ..	13
9-7	CAN Link Level Addressing .....	14
9-8	CAN Frame Types .....	14
9-9	CAN Media Access Control .....	15
9-10	CAN Error Management .....	17
9-10.1	Types of Errors.....	17
9-10.2	Node Error States .....	17
9-11	CAN Chips and DeviceNet . ....	19
9-11.1	Types of CAN Chips.....	19
9-11.2	Number of Acceptance Filters .....	19
9-11.3	Types of Acceptance Filters.....	19
9-11.4	Standard Frame vs. Extended Frame.....	20
9-11.5	Bit Timing Examples .....	20
9-12	CAN Interrupt Rate.....	22

## **9-1 Introduction**

This chapter provides details concerning indicators and switches on DeviceNet products.

## **9-2 Indicators**

Indicators assist maintenance personnel in quickly identifying a problem unit. This is accomplished through the consistent placement and presentation of indicators on DeviceNet products.

This section refers to Indicators that are visible to maintenance personnel. Visible means that:

- Indicators can be viewed without removing covers or parts from the equipment.
- Indicators can be seen in normal lighting.

Any labels and icons must be visible whether or not the Indicator is illuminated.

For indicator requirements specific to safety implementations see CIP Safety Specification (Volume 5, Chapter 9).

### **9-2.1 DeviceNet Indicator Requirements**

DeviceNet does not require a product to have indicators. However, if a product does support any of the indicators described here, they must adhere to the specifications described in this chapter.

Device developers are free to have additional indicators on their products if they so desire.

**Important:** If a product has indicators then it is recommended that either a Module Status LED and a Network Status LED, or the combined Module/Network Status LED be included. These LEDs are very helpful to a user working with the network.

**Important:** If there is a conflict between turning an LED on Red versus Green, the LED should be turned on Red.

## 9-2.2 Module Status LED

This bi-color (green/red) LED provides device status. It indicates whether or not the device has power and is operating properly. Table 9-2.1 and Figure 9-2.1 define the Module Status LED states. The states shown reflect the device states specified in the Identity Object specified in Volume 1, Chapter 5, Object Library.

**Table 9-2.1 Module Status LED**

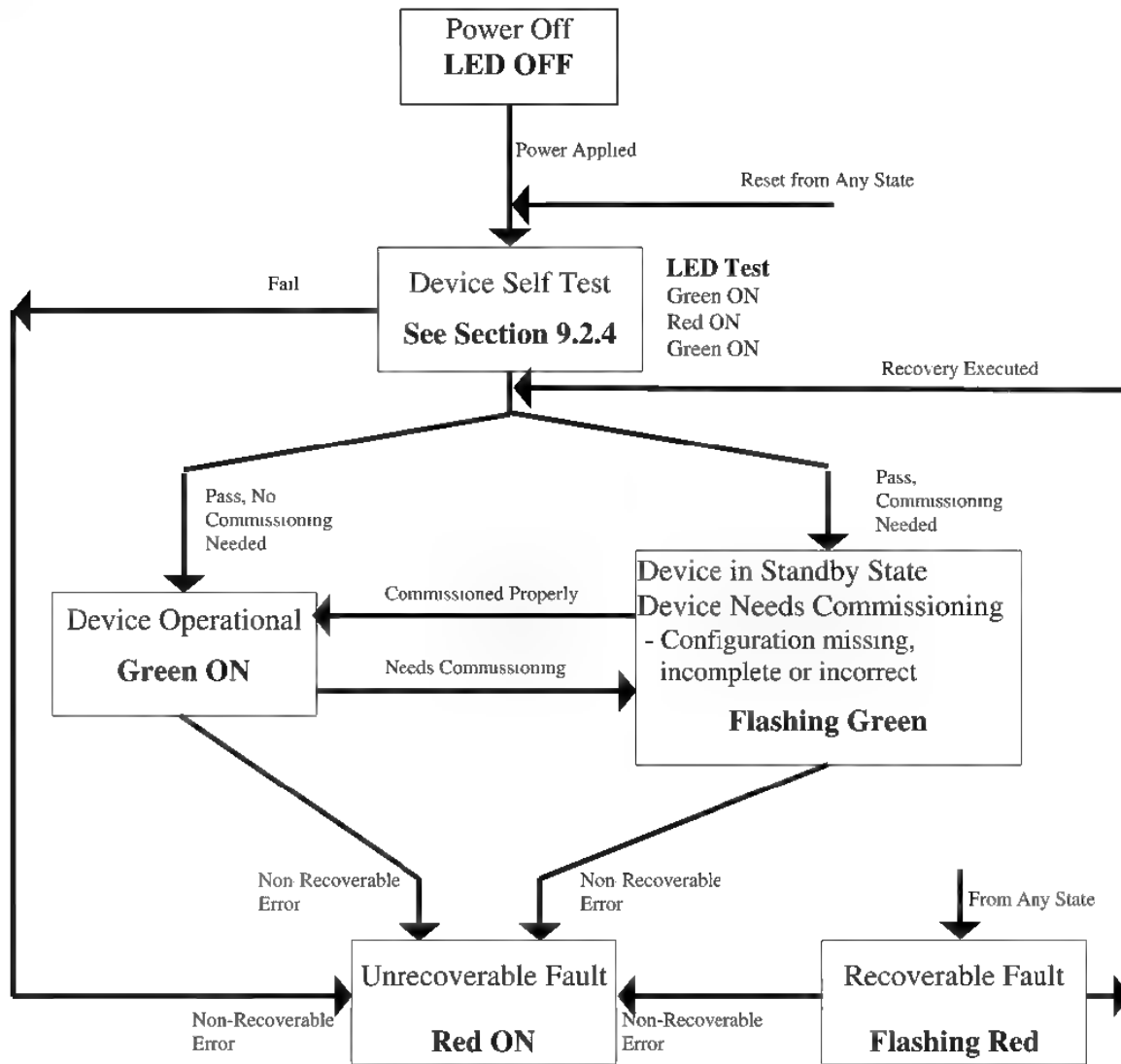
For this state:	LED is:	To indicate:
No Power	Off	There is no power applied to the device.
Device Operational	Green	The device is operating in a normal condition.
Device in Standby (The Device Needs Commissioning)	Flashing Green <sup>1</sup>	The device needs commissioning due to configuration missing, incomplete or incorrect. The Device may be in the Standby state. Reference the Identity Object in Volume 1, Chapter 5: Object Library.
<b>Recoverable Fault</b>	Flashing Red <sup>1</sup>	<b>The device has a recoverable fault.</b>
Unrecoverable Fault	Red	The device has an unrecoverable fault, may need replacing.
Device Self Testing	Flashing Red– Green <sup>1</sup>	The Device is in Self Test. Reference the Identity Object in Volume II for Device states.

<sup>1</sup> For information on LED flash rates, refer to section 9-2.8

Any LED colors/states not defined in Table 9-2.1 are reserved and must not be used.

For information about Module Status LED indications during power-up, refer to section 9-2.4.

Figure 9-2.1 States of the Module Status LED



### 9-2.3 Network Status LED

This bi-color (green/red) LED indicates the status of the communication link. Refer to Chapter 2, *Network Access State Transition Diagram*, to compare the Network Status LED to the Network Access State machine.

Table 9-2.2 defines the Network Status LED states.

**Table 9-2.2 Network Status LED**

For this state:	LED is:	To indicate:
Not Powered/Not On line	Off	Device is not on line - The device has not completed the Dup_MAC_ID test yet. - The device may not be powered, look at Module Status LED. - No network power present.
On-line, Not Connected	Flashing Green <sup>1</sup>	Device is on-line but has no connections in the established state. - The device has passed the Dup_MAC_ID test, is on-line, but has no established connections to other nodes. - For a Group 2 Only device it means that this device is not allocated to a master - For a UCMM capable device it means that the device has no established connections.
Link OK On line, Connected	Green	The device is on line and has connections in the established state. - For a Group 2 Only device it means that the device is allocated to a Master. - For a UCMM capable device it means that the device has one or more established connections.
Connection Time-Out	Flashing Red <sup>1</sup>	One or more I/O Connections are in the Timed-Out state
Critical Link Failure	Red	Failed communication device. The device has detected an error that has rendered it incapable of communicating on the network (Duplicate MAC ID, or Bus-off).
Communication Faulted and Received an Identify Comm Fault Request - Long Protocol	Flashing Red & Green <sup>2</sup>	A specific Communication Faulted device. The device has detected a Network Access error and is in the Communication Faulted state. The device has subsequently received and accepted an Identify Communication Faulted Request - Long Protocol message.

<sup>1</sup> For information on LED flash rates, refer to section 9-2.8

<sup>2</sup> For information on LED flash rates, refer to Chapter 2, section 2-11.4.3

Any LED colors/states not defined in Table 9-2.2 are reserved and must not be used.

For information about Network Status LED indications during power up, refer to section 9-2.4.

#### **9-2.4 Module and Network Status LEDs at Power-Up**

- A LED test is to be performed at power-up. To allow a visual inspection to be performed, the following sequence is to be followed:
- Turn Network Status LED off.
- Turn Module Status LED on Green for approximately 0.25 seconds.
- Turn Module Status LED on Red for approximately 0.25 seconds.
- Turn Module Status LED on Green.
- Turn Network Status LED on Green for approximately 0.25 seconds.
- Turn Network Status LED on Red for approximately 0.25 seconds.
- Turn Network Status LED off.

If a device has other LEDs each LED should be tested in sequence.



## 9-2.5 Combined Module/Network Status LED

This bi-color (green/red) LED provides limited device and communication status. The combined Module/Network (or Mod/Net) Status LED indicates whether or not the device has power and is operating properly. Refer to Chapter 2, *Network Access State Transition Diagram* to compare the combined Module/Network Status LED to the Network Access State machine. Table 9-2.3 illustrates the combined Module/Network Status LED states.

**Table 9-2.3 Combined Module/Network Status LED**

For this state:	LED is:	To indicate:
Device Not Powered/Not On-line	Off	Device is not on-line. - The device has not completed the Dup_MAC_ID test yet. - The device may not be powered.
Device Operational AND On-line, Connected	Green	The device is operating in a normal condition and the device is on-line with connections in the established state. - For a Group 2 Only device it means that the device is allocated to a Master. - For a UCMM capable device it means that the device has one or more established connections.
Device Operational AND On-line, Not Connected or Device On-line AND Device needs commissioning	Flashing Green <sup>1</sup>	The device is operating in a normal condition and the device is on-line with no connections in the established state. - The device has passed the Dup_MAC_ID test, is on-line, but has no established connections to other nodes. - For a Group 2 Only device it means that this device is not allocated to a master. - For a UCMM capable device it means that the device has no established connections. - Configuration missing, incomplete or incorrect.
Minor Fault and/or Connection Time-Out and/or No Network Power	Flashing Red <sup>1</sup>	Any one or more of the following conditions: - Recoverable fault - One or more I/O Connections are in the Timed-Out state - No network power present
Critical Fault or Critical Link Failure	Red	The device has an unrecoverable fault; may need replacing.  Failed communication device. The device has detected an error that has rendered it incapable of communicating on the network (Duplicate MAC ID, or Bus-off)
Communication Faulted and Received an Identify Comm Fault Request - Long Protocol	Flashing Red & Green <sup>2</sup>	A specific Communication Faulted device. The device has detected a Network Access error and is in the Communication Faulted state. The device has subsequently received and accepted an Identify Communication Faulted Request - Long Protocol message.

<sup>1</sup> For information on LED flash rates, refer to section 9-2.8

<sup>2</sup> For information on LED flash rates, refer to Chapter 2-11.4.3

Any LED colors/states not defined in Table 9-2.3 are reserved and shall not be used.

For information about the combined Module/Network Status LED indications during power-up, refer to section 9-2.6.

## 9-2.6 Combined Module/Network Status LED at Power-Up

A LED test is to be performed at power-up. To allow a visual inspection to be performed, the following sequence is to be followed:

- Turn combined Module/Network Status LED on Green for approximately 0.25 seconds.
- Turn combined Module/Network LED on Red for approximately 0.25 seconds
- Turn combined Module/Network LED off.

If a device has other LEDs each LED should be tested in sequence.

## 9-2.7 I/O Status LED

This bi-color (green/red) LED provides information concerning the states of inputs and/or outputs. The terms “inputs” and “outputs” are being applied loosely here. For example, a Pneumatic Valve Pack device developer may model its product using Discrete Output Point Objects.

The intent of the I/O Status LED is to inform the user whether this device has outputs under control and whether any outputs or inputs are active (outputs active, inputs producing, etc.) or faulted. The LED is intended to reflect the mode/state of the inputs and outputs, not necessarily the on/off condition of the I/O points themselves.

The specific definition of the I/O Status LED colors and states reside in the individual object definitions that specify the use of this LED. Refer to the Discrete Output Point and the Discrete Input Point Objects in Chapter 5 of the CIP Common specification for examples. Table 9-2.4 is intended to provide *guidelines* governing the use of the I/O Status LED.

The I/O Status LED should follow the flash rates defined in section 9-2.7.

**Table 9-2.4 I/O Status LED**

For this Output State:	LED is:	To indicate:
Output(s) Inactive Input(s) Inactive	Off	All outputs are inactive. All inputs are inactive.
Output(s) Active Input(s) Active	Green	One or more outputs are active and under control, and no outputs are “faulted”. One or more inputs are active and producing data, and no inputs are “faulted”.
Output(s) Idle	Flashing Green <sup>1</sup>	One or more outputs are idle and no outputs are active nor “faulted”.
Output(s) Faulted Inputs(s) Faulted	Flashing Red <sup>1</sup>	One or more outputs are “faulted”, maybe in the fault state One or more inputs are “faulted”, maybe in the fault state.
Output(s) Forced Off Input Unrecoverable Fault	Red	One or more outputs are forced off (may be an Unrecoverable Fault). One or more inputs has an unrecoverable fault.

<sup>1</sup> For information on LED flash rates, refer to section 9-2.8

Any LED colors/states not defined in Table 9-2.4 are reserved and shall not be used.

### 9-2.8 LED Flash Rate

Unless otherwise indicated, the flash rate of the LED is approximately 1 flash per second. The LED should be on for approximately 0.5 second and off for approximately 0.5 second. This flash rate specification only applies to LEDs specified in this chapter.

## 9-3 Switches

The following rules apply to any switches.

- The device shall provide visual indication (via physical switch, LED, etc) when the MAC ID or Baud Rate switches have been overridden.
- If an attribute is configured with a switch and it cannot be overwritten with software, then the device must respond with an error response to a Set Attribute message using error status code 0E (Attribute not settable).

**Important:** If a product has switches, then it is recommended that the product provide the means by which the switch setting can be determined remotely using messaging services.

For switch requirements specific to safety implementations see CIP Safety Specification (Volume 5, Chapter 9).

### 9-3.1 DeviceNet MAC ID Switches

If DIP switches are used to set the MAC ID they must be in binary format. If rotary, thumbwheel or pushwheel switches are used then decimal format is required. The most significant digit is always to the left or to the top of the product when the user is trying to configure the switches.

### 9-3.2 DeviceNet Baud Rate Switches

If switches are used to set the DeviceNet baud rate they must be encoded as follows:

**Table 9-3.1 DeviceNet Baud Rate Switch Encoding**

For this Baud Rate:	Switch Setting:
125 kbps	0
250 kbps	1
500 kbps	2

## 9-4 Recommended Physical Standards

DeviceNet customers will find it easier to deal with products from multiple vendors if there is consistent labeling of Indicators, Switches and Connectors amongst the DeviceNet products. This section provides some recommendations about how Indicators, Switches and Connectors should be labeled.

### 9-4.1 Recommended Indicator and Switch Labeling

Table 9-4.1 makes provides recommendations concerning how indicators and switches should be labeled on DeviceNet products.

**Table 9-4.1 DeviceNet Indicator and Switch Labeling**

Description	Full Name	Abbreviated Name
Module Status LED	Module Status	MS
Network Status LED	Network Status	NS
Combined Module/Network Status LED	Module/Network Status or, Mod/Net Status	MNS
I/O Status LED	I/O Status or I/O	IO
MAC ID Switches	Node Address	NA
Baud Rate Switches	Data Rate	DR

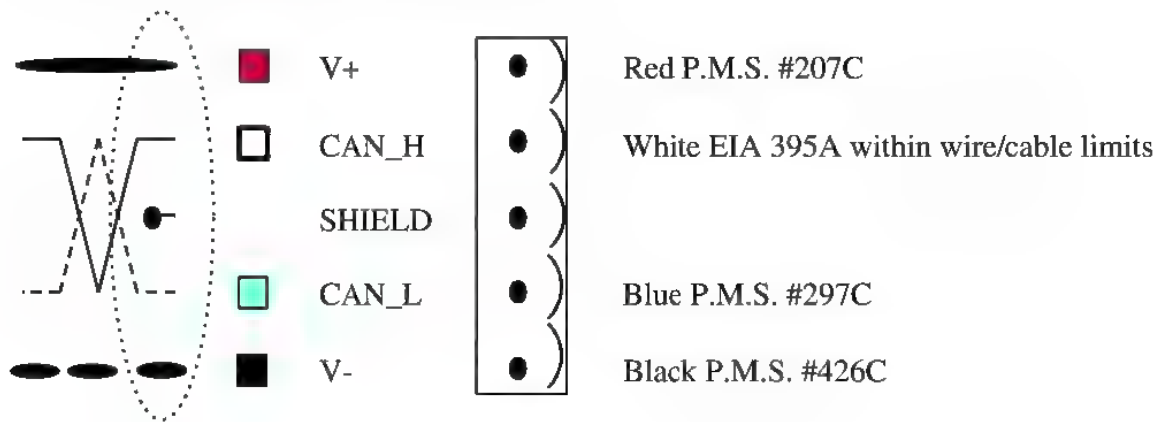
### 9-4.2 Recommended Connector Labeling

This specification makes the following recommendations concerning how connectors should be labeled:

#### 9-4.2.1 Open Style Connector (5-Position)

Figure 9-4.1 shows an icon beside a 5-position open style DeviceNet header. The signal descriptions for each wiring connection are included for clarity, but are not part of the icon. The icon does include a color chip beside each connection, except for the shield, to indicate the wire's insulation color. All except the white use the Pantone Matching System. Pantone does not specify white.

Figure 9-4.1 5-Pin Open Style Connector Icon



### 9-4.3 Recommended Connector Key Orientation

It is recommended that the connector should be keyed such that a DeviceNet cable exits the instrument or device without interfering with any indicators, auxiliary connectors or anything else that may require access in the field. The mating DeviceNet receptacle on the instrument or device should be mounted such that the key orientation allows the cable to have no interference with indicators, auxiliary connectors, or anything else that may require access in the field. See figures 8.3-31, 8.3-33 and 8-3.53 for details.

## 9-5 DeviceNet and CAN

DeviceNet uses the Controller Area Network (CAN) technology. CAN has been accepted for use in automobile applications, and integrated circuits are now available from multiple vendors. This chapter presents a general overview of CAN. Refer to the specifications listed in Chapter 1-6, Normative References, for a full definition of CAN.

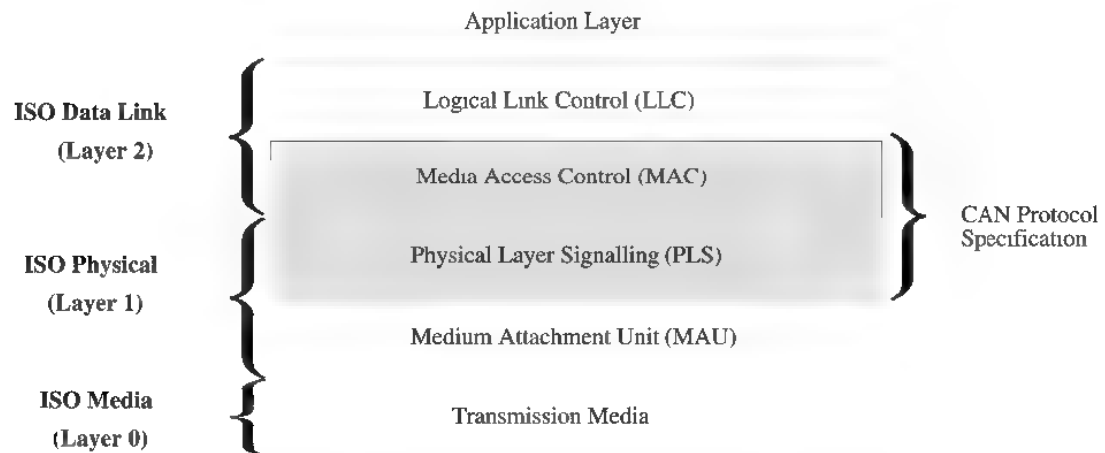
## 9-6 The Scope of CAN

CAN is a communications protocol specification which defines the following:

- A Media Access Control (MAC) methodology
- Physical Signaling

CAN does not specify the entire Physical Layer and/or Medium upon which it resides, or the Application Layer protocol used to move data. Figure 9-6-1 illustrates the scope of the CAN specification.

**Figure 9-6-1 Scope of CAN**



## 9-7 CAN Link Level Addressing

CAN is a broadcast oriented protocol. The various frames transmitted on the network are assigned an *identifier* and each station decides, based on this identifier, whether or not it receives the frame. This identifier is specified in the *Identifier Field* of a CAN frame.

## 9-8 CAN Frame Types

The following frame types are defined by CAN:

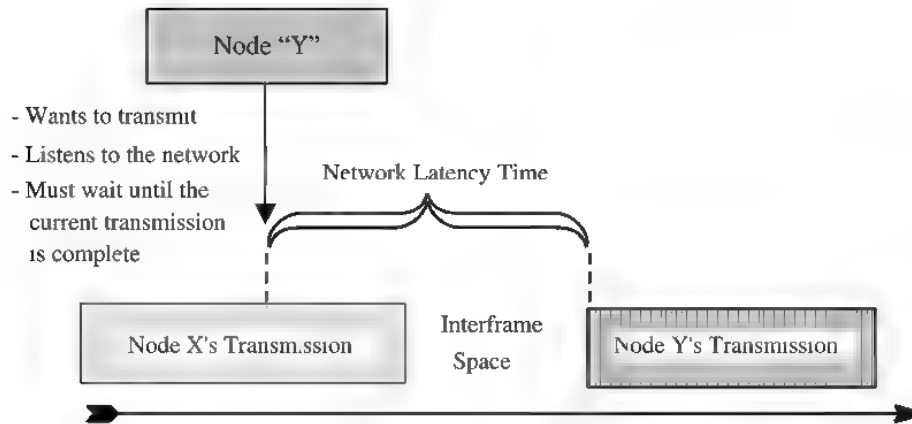
- **Data Frame** - Moves data from a transmitter to the receiver(s).
- **Remote Frame** - Requests transmission of the Data Frame associated with the specified identifier.
- **Error Frame** - Signifies that a node has detected a bus/network error. Two types of the Error Frame exist, based on the “state” of the node (see section 9-10.2, Node Error States, on page 17).
- **Overload Frame** - Provides a delay between the transmission of frames to control the flow of data.

**Important:** DeviceNet does not make use of the CAN Remote Frame.

## 9-9 CAN Media Access Control

A node's transmission on CAN is heard and acknowledged by ALL other nodes on the network. Whenever the bus is free of transmission, a node can begin to transmit. If a node is transmitting, this transmission must be completed before another node can attempt to transmit. See Figure 9-9-1.

Figure 9-9-1 CAN Media Access



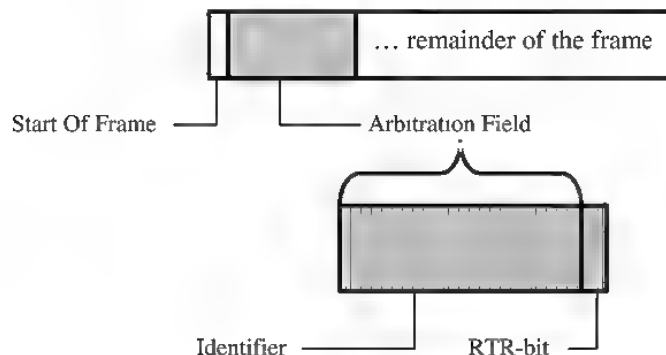
If two or more nodes begin transmitting at the same time, the conflict is resolved by a non-destructive bitwise arbitration algorithm using the **Arbitration Field**. The Arbitration Field is included in all CAN Data Frames. The *Arbitration Field* comprises:

- the 11-bit CAN Identifier Field
- RTR-bit

The **RTR-bit** indicates whether the frame is an actual Data Frame or a Remote Frame (see section 9-8, CAN Frame Types).

Figure 9-9-2 illustrates the placement of the Arbitration Field within a CAN Frame.

Figure 9-9-2 Arbitration Field





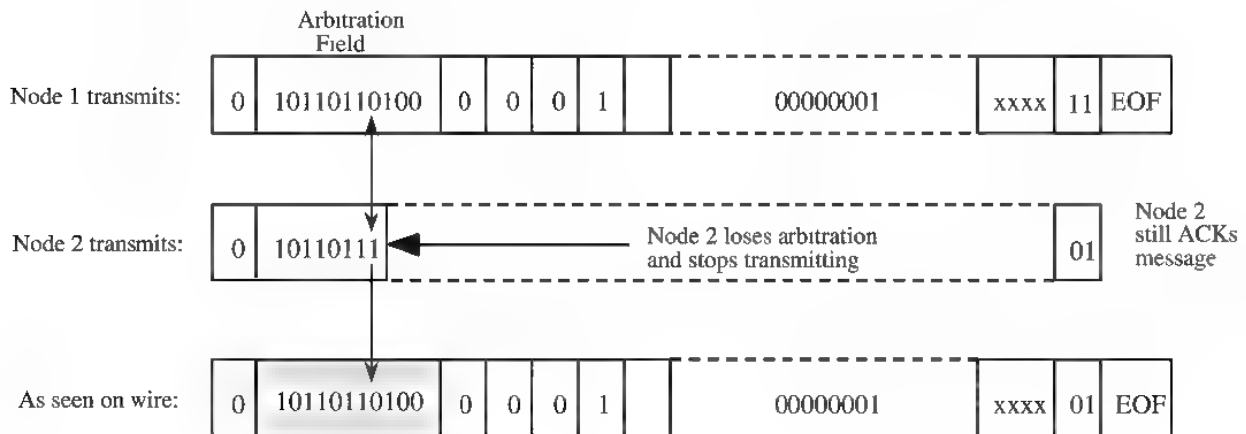
The 11-bit Identifier Field is transmitted from most significant to least significant bit. A bit on the bus can be either dominant (value 0) or recessive (value 1). Simultaneous transmission of a dominant bit and a recessive bit results in the presence of a dominant bit. Refer to Chapter 8-1.3 for information concerning DeviceNet's representation of the dominant and recessive levels.

**Important:** Since DeviceNet does not make use of the CAN provided Remote Frame, the RTR-bit is always *dominant*.

During transmission of the Arbitration Field, every transmitter monitors the current level on the bus and compares it with the bit it has transmitted. If the values are equal, then the node is able to continue the transmission. If a recessive bit (value 1) was transmitted and a dominant bit (value 0) is monitored on the bus, then the transmitting node has lost the arbitration sequence and must discontinue the transmission without sending any more data. The node that lost arbitration can attempt the transmission again when the current transmission is complete.

**Important:** The identifier with the lowest value wins the arbitration sequence.

**Figure 9-9-3 Example of Bitwise Arbitration**



## 9-10 CAN Error Management

### 9-10.1 Types of Errors

CAN provides mechanisms to detect the following types of errors:

- **Bit Error** - Occurs when a transmitter compares the level on the bus with the level it transmitted and the two are not equal. The detection of a dominant bit when a recessive bit was transmitted does not result in a bit error during transmission of the Arbitration Field, the ACK Slot, or a Passive Error Flag.
- **Acknowledgment Error** - Occurs when the transmitter determines that the message has not been acknowledged. An Acknowledgment Slot exists within both Data Frames and Remote Frames. Within this slot, *ALL RECEIVING NODES, regardless of whether or not they are the intended recipient*, must acknowledge the receipt of a message.
- **Stuff Error** - Occurs when a node detects six (6) consecutive bits of the same value. In normal operation, when a transmitter detects that it has sent five (5) consecutive bits of the same value, it *stuffs* the next bit with the opposite value (this is called *bit stuffing*). All receivers remove stuffed bits prior to the calculation of the CRC (*Cyclic Redundancy Check*). Thus, when a node detects six (6) consecutive bits of the same value, a stuff error has occurred.
- **CRC Error** - Occurs when the CRC (*Cyclic Redundancy Check*) value does not match the value generated by the transmitter. Every frame contains a *Cyclic Redundancy Check* (CRC) field which is initialized by the transmitter. The receivers calculate a CRC and compare it against the value the transmitter generated. If the two values don't match, a CRC error has occurred.
- **Form Error** - Occurs when an invalid bit value is detected in an area where a predefined value must be transmitted. Certain predefined bit values exist that must be transmitted at certain points within a CAN frame. If an invalid bit value is detected in one of these areas, a form error has occurred.

### 9-10.2 Node Error States

To minimize the negative effects a faulty node has on the network, and, consequently, to provide fault confinement, CAN defines a fault confinement state machine. A node can be in one of the following three error states:

- **Error Active** - When an *error active* node detects one of the above errors it transmits an *Error Active Frame*, which consists of six (6) consecutive dominant bits. This transmission will override any other transmission happening at the same time and will cause all other nodes to detect a *stuff error*, which in turn, causes them to discard the current frame.

When a node in the *error active* state detects a problem with a transmission, the node prevents all the others from receiving the packet by transmitting an *Error Active Frame*. This process is performed regardless of whether or not the node detecting the error was the intended recipient of the data.

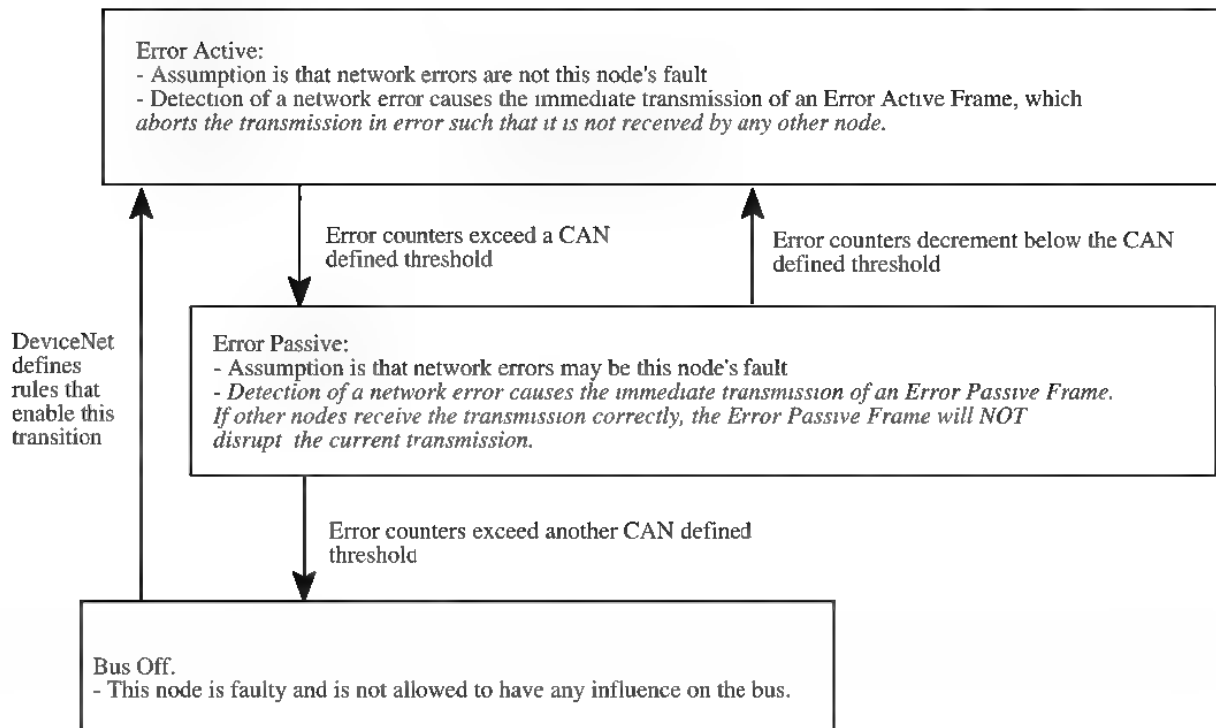
- **Error Passive** - When an *error passive* node detects one of the above errors it transmits an *Error Passive Frame* which consists of six (6) consecutive recessive bits. This frame may be overwritten by a transmission occurring at the same time and does not cause a frame to be discarded if the other stations do not detect an error.

- **Bus Off** - A node in the *bus off* state is not allowed to have any influence on the bus. It is logically disconnected from the network.

A general overview of the transitions involved in the fault confinement state machine is presented below:

- A node keeps track of transmit and receive error counters.
- A node starts out in the *error active* state with its error counters equal to zero (0). A node in this state assumes any error detected is not its fault.
- The type of error and the point at which they are detected are assigned different “count values” which are added to the running total depending on whether the error is a transmit or receive error. Valid receptions and transmissions cause these counters to decrement, with zero (0) being the minimum value.
- When either of these counters passes a CAN defined threshold, the node enters the *error passive* state. In this state the node *suspects* that it is the cause of the error.
- When the transmit error counter passes yet another CAN defined threshold, the node enters the *bus off* state. This specification defines mechanisms for enabling the transition from *bus off* to *error active*.
- When both the transmit and receive counters of an *error passive* node decrement below the CAN defined threshold, the node is once again *error active*.

Figure 9-10-1 CAN Error States



## 9-11 CAN Chips and DeviceNet

CAN chips which implement the CAN protocol described in the preceding sections are available from multiple vendors. Product designers must be careful in the selection of a CAN chip. You need to fully understand how your product will use DeviceNet and what the processing needs of your application are. Furthermore, you need to understand how DeviceNet operates on CAN, how CAN functions, and how CAN chips operate.

CAN chips are available with various capabilities, some of which are described in the following sections.

### 9-11.1 Types of CAN Chips

The following two types of CAN chips exist:

- Embedded - Chips that include the CAN controller and some form of integrated microcontroller.
- Peripheral - Chips that contain just the CAN controller.

### 9-11.2 Number of Acceptance Filters

Acceptance Filters can be thought of as sieves, or identifier screeners. Any message that gets through the Acceptance Filters must be handled by the processor servicing the CAN controller. The more items of non interest that can be filtered, the less the impact to the processor.

CAN chips may have a single filter or multiple filters, depending on the chip.

### 9-11.3 Types of Acceptance Filters

The following two types of filters exist within CAN chips:

- Fixed - filters that require the bits to match exactly, one-for-one.
- Mask-and-Match (Mask/Match) - filters that apply a mask to the identifier field before it is compared to the Acceptance Code Register, allowing bits to become don't care bits.

For example; In Figure 9-11-1 the *Mask Register* is configured such that the received identifier bits 10-6 must match bits 10-6 in the *Acceptance Code Register*. In this example, the received identifier bits 10-6 must be 11110 and the rest of the bits are don't care. If bits 10-6 are 11110 then this message is accepted, regardless of the values in bits 5-0.

**Figure 9-11-1 Mask-and-Match Acceptance Filtering**

Mask and Match Acceptance Filtering											
10	9	8	7	6	5	4	3	2	1	0	Identifier Bit Positions
1	1	1	1	0	0	0	0	1	1	1	Identifier Bits Received
mm	mm	mm	mm	mm	X	X	X	X	X	X	Mask Register
1	1	1	1	0	0	0	0	0	0	0	Acceptance Code Register
This message is accepted											

Mm = must match the acceptance code

x = don't care, accept a '1' or a '0'

#### 9-11.4 Standard Frame vs. Extended Frame

CAN chips can support either the *Standard Frame* or the *Extended Frame*.

**Standard Frame** means the CAN chip supports an 11-bit Identifier Field. **Extended Frame** means the CAN chip supports a 29-bit Identifier Field. Recent CAN chips support both Standard Frame and Extended Frame formats.

**Important:** DeviceNet is based solely on the Standard Frame (11 bit Identifier Field) definition of CAN.

#### 9-11.5 Bit Timing Examples

Examples in this section refer specifically to the Philips 82C200, Intel 82527, and Motorola 68HC05X4 CAN controllers. Examples presented in Table 9-11.1 and Table 9-11.2 illustrate both 16 and 20 MHz clocks. Other clock rates can be used, provided that the maximum allowable cable delay is not decreased. This means that the sampling point must occur no sooner than one sys clock after 0.80 (80%) of the nominal bit time. Output control in the examples sets TX1 = data with dominant high polarity. DeviceNet requires a bit time accuracy of +/- 1000 ppm.

**Table 9-11.1 Configuration Requirements**

Configuration	Specification
Control Register	Set to sync on one edge.
BTR0	SJW = 0 (16 MHz) SJW = 1 (20 MHz)
prescaler	= 3 (+4) @ 125 Kbaud = 1 (+2) @ 250 Kbaud = 0 (+1) @ 500 Kbaud
BTR1:	one sample per bit
	TSEG1(16 MHz): set to provide 13 sys. clocks TSEG1(20 MHz): set to provide 16 sys. clocks
	TSEG2(16 MHz): set to provide 2 sys. clocks TSEG2(20 MHz): set to provide 3 sys. clocks
Suggested Output Control	Polarity: TX0 and RX0 - dominant is low
	Tx Drive - push-pull
	RX1-enabled (external VDD/2 reference)
	TX1-disabled or other output if desired
Output Clock (if used)	Output at 16 or 20 MHz, depending on xtal

**Table 9-11.2 Configuration Examples**

<b>Examples:</b>		<b>82527 @ 16 MHz</b>	<b>82C200 @ 16 MHz</b>	<b>68HC05X4 @ 16 MHz</b>	<b>68HC05X4 @ 20 MHz</b>
<b>BTR0</b>	(125KB)	0x03	0x03	0x03	0x43
	(250KB)	0x01	0x01	0x01	0x41
	(500KB)	0x00	0x00	0x00	0x40
<b>BTR1</b>	(125KB)	0x1C	0x1C	0x1C	0x2F
	(250KB)	0x1C	0x1C	0x1C	0x2F
	(500KB)	0x1C	0x1C	0x1C	0x2F
Bus Configuration		0x08	n/a	n/a	n/a
Clock Output		0x20	n/a	n/a	n/a
Output Control		n/a	0xFA	0xFA	0xFA

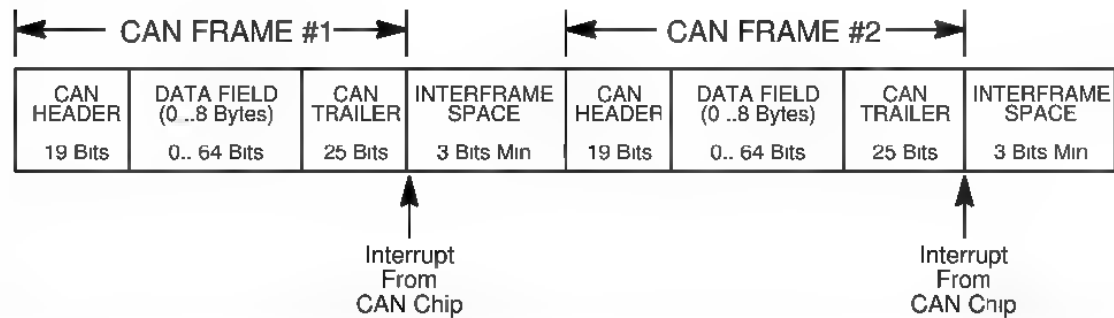
## 9-12 CAN Interrupt Rate

Designers should consider the interrupt rate from their CAN controllers when designing DeviceNet products. Because the CAN data frame is small (0 to 8 bytes), the occurrence of interrupts to the processor may be high. In this respect, do not consider CAN a low-speed network.

Figure 9-12-1 shows two back-to-back CAN Data Frames with minimal spacing between the frames (called Interframe Spacing). Table 9-12.1, created from the figure, shows the **worst-case** interrupt rate from a CAN receiver, which is receiving all frames on a flooded (continuous frames back-to-back) link.

The “Number of CAN Bits” row in the table assumes no bit stuffing (bit stuffing would increase the time between interrupts). While it may be difficult to have a continuously flooded link, there certainly will be times when many devices will want to send messages at the same time, causing bursts of back-to-back frames. Any device screening in software may see the interrupt rates shown in Figure 9-12-1 and Table 9-12.1.

**Figure 9-12-1 CAN Data Frame Interrupts**



**Table 9-12.1 Minimum CAN Interrupt Times Per DeviceNet Frame**

Number of Data Bytes	0	1	2	3	4	5	6	7	8
Number of CAN Bits	47	55	63	71	79	87	95	103	111
Interrupt Time at 125 kbps ( 1 Bit Time = 8 $\mu$ S )	376	440	504	568	632	696	760	824	888
Interrupt Time at 250 kbps ( 1 Bit Time = 4 $\mu$ S )	188	220	252	284	316	348	380	412	444
Interrupt Time at 500 kbps ( 1 Bit Time = 2 $\mu$ S )	94	110	126	142	158	174	190	206	222

\*All Interrupt Times are expressed in microseconds ( $\mu$ S). Information is presented assuming no stuff bits

As the table indicates, *DeviceNet interrupt rates are NOT low speed!* At 500 kbps, interrupts can occur every 94  $\mu$ s for zero byte data frames. Most low-end microcontrollers cannot keep up with this interrupt rate. Therefore, a trade-off between CAN Controller capability and Cost/Package size exists.

Select CAN Controllers that provide the proper level of Acceptance Filtering. The controller should have sufficient processing time left to run the application, and/or dedicate a separate microcontroller/microprocessor to service the CAN receiver.

Also, be aware that some CAN chips screen only the most significant eight bits of the Identifier Field (not the entire 11 bits) and have one Mask/Match filter. Such chips include Motorola's MC68HC05X4/X16 family and Philips' p82C200 and p82C592 chips.



This page is intentionally left blank.

## **Volume 3: DeviceNet Adaptation of CIP**

# **Chapter 10: Bridging & Routing**

---

## Contents

10-1	Introduction.....	3
10-2	Unconnected Explicit message routing .....	3
10-2.1	Unconnected Send Service Routing as Intermediate Hop.....	3
10-2.1.1	Unconnected Send Service Request Modification .....	3
10-2.1.2	Unconnected Send Service Response Modification.....	4
10-2.2	Unconnected Send Service Routing as Last Hop .....	5
10-2.3	Unconnected Send Service Routing Examples .....	5
10-2.3.1	Unconnected Send Service Routing Example (From DeviceNet).....	5
10-2.3.2	Unconnected Send Service Routing Example (Intermediate and Last Hop).....	9
10-3	Connected Explicit Message Routing to DeviceNet .....	12
10-3.1	Explicit Message Connection Establishment to a DeviceNet Target .....	12
10-3.2	Explicit Message Connection - Request.....	12
10-3.3	Explicit Message Connection - Response .....	13
10-3.4	Explicit Message Connection Establishment Example (Last Hop).....	13
10-4	I/O Message Routing to DeviceNet .....	15
10-4.1	I/O Connection Establishment to a DeviceNet Target .....	15
10-4.1.1	Verify Connection Parameters .....	16
10-4.1.2	Electronic Key Verification .....	16
10-4.1.3	Multicast Production at Target Check.....	17
10-4.1.4	Dynamic I/O Connection Creation Attempt.....	17
10-4.1.5	Second Dynamic I/O Connection Creation, if Required .....	17
10-4.1.6	Mapping Forward Open Connection Parameters .....	17
10-4.1.7	Connection ID Selection .....	19
10-4.1.8	Handling Configuration Data in Forward Open .....	19
10-4.1.9	Process the Production Inhibit Time Network Segment .....	19
10-4.1.10	Establishing the Dynamic I/O Connection with the ApplyAttributes Service .....	19
10-4.1.11	Predefined Master/Slave Connection Set allocation .....	20
10-4.1.12	Verification of Connection Timeout Multiplier .....	20
10-4.1.13	Verification of Predefined Connection Paths.....	20
10-4.1.14	Handling Configuration Data to a Group 2 Server.....	20
10-4.1.15	Establishing the Predefined I/O Connection by Setting the EPR Attribute.....	21
10-4.1.16	Returning a Success Response .....	21
10-4.2	I/O Connection Usage.....	21
10-4.3	I/O Connection Establishment Example .....	21
10-5	Connected Message Routing Through/From DeviceNet .....	24
10-5.1	I/O and Explicit Message Routing Through DeviceNet (Intermediate Hop).....	24
10-5.2	I/O and Explicit Message Routing From DeviceNet.....	24
10-5.3	I/O Message Routing Through DeviceNet Example (Intermediate Hop) .....	24
10-5.4	Explicit Message Routing from DeviceNet Example .....	27
10-6	Closing Connections .....	30
10-6.1	Using the Forward Close Service to Close an I/O Messaging Connection .....	30

## **10-1 Introduction**

DeviceNet connection establishment differs from the general CIP model. In particular, the services of the Connection Manager are not supported by DeviceNet nodes when they are the target of the request. Thus, only devices that provide routing to and/or from DeviceNet support the Connection Manager object, and are required to do so.

When the target node of a routed message is on DeviceNet, the DeviceNet router is required to translate the message into normal DeviceNet explicit messaging format. As a result, DeviceNet nodes require no changes to accept routed messages from other CIP subnets.

Routed messages consist of Explicit Message routing using the Unconnected Send service; Explicit Message routing to DeviceNet using the Forward Open service; I/O message routing to DeviceNet; and both Explicit and I/O message routing from/through DeviceNet. This chapter will describe the methods used to provide bridging and routing of these messages on a DeviceNet subnet.

## **10-2 Unconnected Explicit message routing**

Unconnected explicit message routing is accomplished using the Unconnected Send service. There are two different scenarios concerning the reception of an Unconnected Send service by a DeviceNet router: 1) Routing the service to/from DeviceNet as an intermediate hop; 2) Routing the service to DeviceNet as the last hop. In both of these cases the establishment of a DeviceNet explicit message connection is required (if one is not already established to that node).

### **10-2.1 Unconnected Send Service Routing as Intermediate Hop**

A DeviceNet router shall forward a received Unconnected Send service request to the destination routing node on the subnet when acting as an intermediate hop. In order to allow DeviceNet routers the ability to handle multiple outstanding transactions on DeviceNet, the Unconnected Send service of the Connection Manager is modified when traversing a DeviceNet subnet.

#### **10-2.1.1 Unconnected Send Service Request Modification**

When sent on a DeviceNet subnet, the Unconnected Send service data is prepended with a 16-bit transaction ID. This transaction identifier is generated by the requesting (source) routing device and returned by the responding (destination) routing device along with the response from the target. The modified service request is as shown below.

**Table 10-2-1 Unconnected Send Service Parameters**

Parameter Name	Data Type	Description
Transaction_ID	UINT	Used for transaction management in the DeviceNet Requesting Device and DeviceNet Routers. This field is used for transaction matching by message originators on DeviceNet and not passed on to other subnets
Priority/Time_tick	BYTE	Used to calculate request timeout information.
Time-out_ticks	USINT	Used to calculate request timeout information.
Message_Request_Size	UINT	Specifies the number of bytes in the Message Request.
Message_Request <sup>1</sup>	Struct of	
Service	USINT	Service code of the request.
Request_Path_Size	USINT	The number of 16 bit words in the Request_Path field (next element).
Request_Path	Padded EPATH	This is an array of bytes whose contents convey the path of the request (Class ID, Instance ID, etc.) for this transaction.
Request_Data	Array of octet	Service specific data to be delivered in the Explicit Messaging Request. If no additional data is to be sent with the Explicit Messaging Request, then this array will be empty.
Pad	USINT	Only present if Message_Request_Size is an odd value.
Route_Path_Size	USINT	The number of 16 bit words in the Route_Path field.
Reserved	USINT	Reserved byte. Shall be set to zero (0).
Route_Path	Padded EPATH	Indicates the route to the Remote Target Device.

<sup>1</sup> This is the Message Router Request Format as defined in Volume 1, Chapter 2-4.

### 10-2.1.2 Unconnected Send Service Response Modification

A DeviceNet router shall always return a successful response on DeviceNet to an Unconnected Send service request. This success response may actually indicate that an error was encountered. This is necessary because additional error information is needed to handle routing errors and this additional information does not conform to the Error Response format defined for DeviceNet. As specified in Volume 1, Chapter 3-5.5.4, the Service code returned shall be either the service code sent inside the Unconnected Send service data *or* the Unconnected Send service code. The 0x94 Error Response Service is never returned.

The modified successful and unsuccessful service responses are as shown below. Note that this is the data placed within the DeviceNet Service Data area; the reply service code is earlier in the packet.

**Table 10-2-2 Successful Unconnected Send Response**

Parameter Name	Data Type	Description
Transaction_ID	UINT	Echoes the value received in the associated Unconnected Send Request.
General Status	USINT	This value is zero (0) for successful transactions.
Reserved	USINT	Shall be zero.
Service Response Data	Array of byte	This field contains the Explicit Messaging Service Data returned by the Target Device/Object. For example, this would contain Attribute Data in response to a Get_Attribute_Single request. If the Explicit Messaging response returned by the Target Device/Object did not contain any Service Data, then this field shall be empty.

The response Service Data associated with an unsuccessful Unconnected Send response is defined below.

**Table 10-2-3 Unsuccessful Unconnected Send Response**

Parameter Name	Data Type	Description
Transaction_ID	UINT	Echoes the value received in the associated Unconnected Send Request.
General Status	USINT	One of the General Status codes listed in Volume 1, Appendix B, Status Codes. If a routing error occurred, it shall be limited to the values specified in the Routing Error Values table.
Size of Additional Status	USINT	Number of 16 bit words in Additional Status array.
Additional Status	Array of UINT	When returning an error from a target that is a DeviceNet node, the Additional Status shall contain the 8 bit Additional Error Code from the target in the lower 8 bits and a zero (0) in the upper 8 bits.
Remaining Path Size	USINT	This field is only present with routing type errors and indicates the number of words in the original route path ( <i>Route_Path</i> parameter of the Unconnected Send Request) as seen by the router that detects the error.

## 10-2.2 Unconnected Send Service Routing as Last Hop

As with other CIP subnets, when a DeviceNet subnet is the last hop the router shall deliver the explicit request contained within the Unconnected Send service to the target node rather than the Unconnected Send service. Since the DeviceNet UCMM does not provide a method for explicit requests, the router shall use a newly created or existing Explicit Message connection to the target for message delivery.

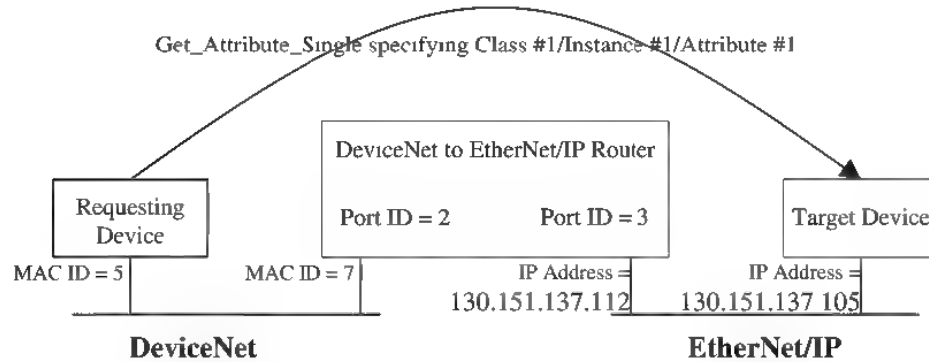
## 10-2.3 Unconnected Send Service Routing Examples

The unconnected routing scenarios presented in this section are illustrated in the following examples.

### 10-2.3.1 Unconnected Send Service Routing Example (From DeviceNet)

Assume that the *Requesting Device* wants to execute a *Get\_Attribute\_Single* on Class #1/Instance #2/Attribute ID #3 in the *Target Device*.

Figure 10-2-1 Single Hop Unconnected Send (Explicit Message Request/Response)



The table below presents the contents of the Service Data field for the Unconnected Send request which is used to execute the Get\_Attribute\_Single example noted above (all values are in hex).

Table 10-2-4 Single Hop Unconnected Send Request Service Data

Bytes	Meaning
01 00	The Requesting Device generated Transaction_Id field.
pp	Priority/Tick Time field.
tt	Connection Timeout Ticks field.
0C 00	Message_Size field = 12 bytes. Note that the Request_Data field is empty.
0E	Service field = Get_Attribute_Single.
05	Request_Path_Size field indicates that there are 5 words in the path field.
21 00 01 00 25 00 02 00 30 03	Path field specifying 16 bit Class Id = 1, 16 bit Instance ID = 2, 8 bit Attribute Id = 3. Note the presence of the pad bytes immediately following the 0x21 and 0x25 bytes. This path could have also been formatted as follows: 20 01 24 02 30 03 which makes use of 8 bit values to convey the Class and Instance ID data. In the 8 bit case, the pad bytes are not required.
09	Route_Path_Size field indicates that there are 9 words in the route path.
00	Reserved byte.
13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00	Contents of the Route_Path field. This field specifies the routing information. These bytes indicate that the request is to be delivered out Port #3 and to IP Address 130.151.137.105. This path encoding uses the Extended Link Address format for the IP address.

#### **Step 1 – Request delivered to the DeviceNet to EtherNet/IP Router**

The first step in the process calls for the Requesting Device to establish an Explicit Messaging Connection with the DeviceNet to EtherNet/IP Router and issue the Unconnected Send request. The table below presents the entire Unconnected Send request. Assume that the Message Body Format established for the Explicit Messaging Connection is DeviceNet 8/8.

**Table 10-2-5 Explicit Message Request on DeviceNet**

Bytes	Meaning
07	Frag = 0 <sup>1</sup> , Transaction ID = 0, Destination MACID = 7.
52	Service = 52. In this context, this specifies the Unconnected Send service.
06	Class ID of the Connection Manager Object Class.
01	The Instance ID Field specifying instance 1 of the Connection Manager.
01 00 pp tt 0C 00 0E 05 21 00 01 00 25 00 02 00 30 03 09 00 13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00	Unconnected Send Service Request Data. This is the previously described Unconnected Send service data describe above with the Transaction_Id field as the first UINT value (01 00 for this example).
<sup>1</sup> - This request would have to be fragmented to deliver it to the router. Fragmentation is not illustrated in this example.	

The Unconnected Send request has now been delivered to the DeviceNet to Ethernet/IP Router via an Explicit Messaging Connection that exists between the Requesting Device and the DeviceNet to Ethernet/IP Router.

### **Step 2 – Router delivers the request to the Target Device**

When the Connection Manager Object within the DeviceNet Router receives the Unconnected Send request it examines the contents of the *Route\_Path* field. In this case, the contents are “13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00”. This indicates that the next device in the hop is on Port #3 and its Node (IP) Address is 130.151.137.105. Additionally, since there is only a single Port/Node Address pair specified in the *Route\_Path*, the request has reached its destination network. When a DeviceNet Router receives an Unconnected Send Request that DOES NOT need to be forwarded through more intermediate CIP Routers, the following steps are taken:

- The DeviceNet Router executes the timing related logic associated with the Priority/Tick Time and Connection Timeout Ticks fields. If a timeout is detected, then a successful Unconnected Send response that specifies a Routing Error is returned to the Requesting Device.
- The DeviceNet Router extracts the actual transaction from the Message Request portion of the Service Data and delivers the Explicit Messaging Request to the Target Device. The actual method of delivering the explicit message is dependant on the link type.

In this example, the message request inside the Unconnected Send indicates a Get\_Attribute\_Single to Class #1, Instance #2, Attribute #3. This approach has no effect on the Target Device. The Target Device does not even realize it has just responded to a request that originated on a remote DeviceNet.

### **Step 3 – Target returns the Explicit Message response to the Router**

The Target Device processes the explicit message and returns a response to the Router. When the Get\_Attribute\_Single Response is received by the DeviceNet Router it generates the Unconnected Send Response and internally delivers it to the Explicit Messaging Connection across which the Unconnected Send Request was originally received.



#### Step 4 – Router returns the Unconnected Response to the Requesting Device

The DeviceNet Router then delivers the Unconnected Send Response to the original Requesting Device.

Assume that the Target Device returns a successful response to the Get\_Attribute\_Single and the requested attribute is a UINT whose value is 0x1234. The text below presents the contents of the CAN Data Field associated with the Unconnected Send Response returned to the Requesting Device:

**Table 10-2-6 Successful Explicit Message Response on DeviceNet**

Bytes	Meaning
05	Frag = 0, Transaction ID = 0, Destination MAC = 5.
8E	Service = 8E. In this context, this specifies a response to the Unconnected Send service.
01 00	The Transaction_Id field echoed back in the response.
00	The General Status field. The value zero (0) indicates that there were no routing errors.
00	The Success status has no additional status.
34 12	The Target Device's Explicit Messaging Response Service Data field. In this case, the value 0x1234 was returned in the Get Attribute Single response.

Now assume that the Target Device returned an Error Response whose General Error Code was set to the value “2” and whose Additional Error Code field was set to the value “5”. The text below presents the contents of the CAN Data Field associated with the Unconnected Send Response returned to the Requesting Device. The absence of the *Remaining Path Size* field indicates that this error (Error Code = 2) was returned from the Target Device, not an intermediate router.

**Table 10-2-7 Unsuccessful Explicit Message Response on DeviceNet**

Bytes	Meaning
05	Frag = 0, Transaction ID = 0, Destination MAC = 5.
8E	Service = 0x8E. In this context, this specifies a response to the Unconnected Send service.
01 00	The Transaction_Id field echoed back in the response.
02	The General Status field. The non-zero value indicates there was an error.
01	The Size of Additional Status field. This indicates that the Target Device returned 16 bits of additional status.
05 00	This indicates that the Target Device returned an Additional Error Code of 0x05.

Finally, assume that the DeviceNet Router was unable to establish an Explicit Messaging Connection with the Target Device. In this case a Routing Error has occurred (the request was not delivered to the Target Device). The error code and extended status that most closely conveys the routing error is: General Code = 0x01, Extended Code = 0x0204 – *Unconnected Send timed out waiting for a response* (Note that error handling is dealt with in more detail in the Event/Action Matrix in Volume 1, Chapter 10-4). The text below presents the contents of the CAN Data Field associated with the Unconnected Send Response returned to the Requesting Device:

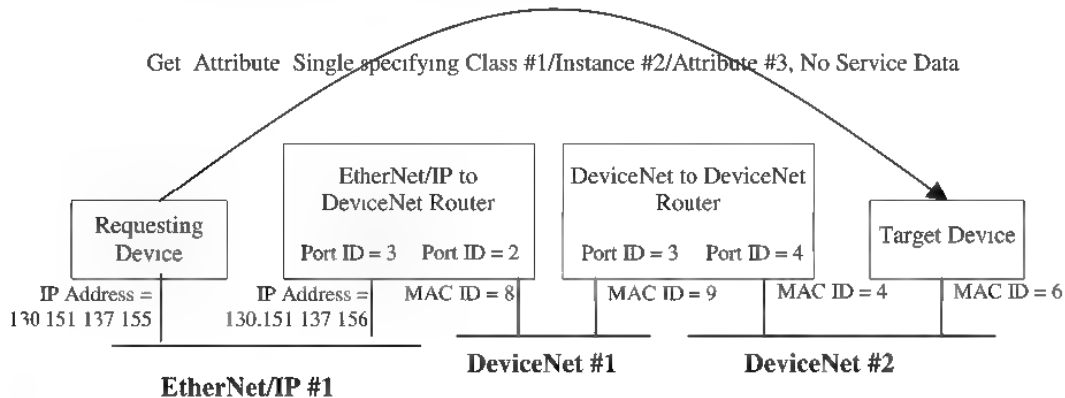
**Table 10-2-8 Routing Failure Explicit Message Response on DeviceNet**

Bytes	Meaning
05	Frag = 0, Transaction ID = 0, Destination MAC = 5.
D2	Service = D2. In this context, this specifies a response to the Unconnected Send service.
01 00	The Transaction_Id field echoed back in the response.
01	The General Status field. This indicates that a routing error was detected.
01	The Size of Additional Status field. This indicates that there is 16 bits of additional status.
04 02	The Additional Status field. There is a single UINT whose value is 0x0204.
00	The <i>Remaining PathSize</i> field. The value zero (0) indicates that an error was encountered during the attempt to establish communications with the Remote Target Device versus another DeviceNet Router.

### 10-2.3.2 Unconnected Send Service Routing Example (Intermediate and Last Hop)

Assume that the Requesting Device wants to execute a `Get_Attribute_Single` on Class #1/Instance #2/Attribute ID #3 in the Target Device. No service data is sent in this request. Notice that this transaction will flow through two CIP Routers.

**Figure 10-2-2 Multiple Hop Unconnected Send**



#### **Step 1 – Request delivered to the EtherNet/IP to DeviceNet Router**

The first step in the process calls for the Requesting Device to issue the Unconnected Send request containing the service request to the EtherNet/IP to DeviceNet Router. The text below presents the contents of the Unconnected Send request's Service Data field which is used to execute the `Get_Attribute_Single` noted above (all values are in hex).

**Table 10-2-9 Multiple Hop Unconnected Send Request Service Data (First Hop)**

Bytes	Meaning
07	Priority/Tick Time field (Time Tick = 128 ms).
0C	Connection Timeout Ticks field (12 Ticks which results in 1536 ms timeout).
08 00	Message_Size field = 8 bytes. Note that this means that no pad byte will be inserted between the Request_Data field and the Route_Path_Size field.
0E	Service field = Get_Attribute_Single.
03	Request_Path_Size field indicates that there are 3 words in the request path.
20 01 24 02 30 03	Request_Path field specifying 8 bit Class Id = 1, 8 bit Instance ID = 2, 8 bit Attribute Id = 3.
	Request_Data field empty
02	Route_Size field indicates that there are 2 words in the Route_Path.
00	Reserved byte.
02 09 04 06	Contents of the Route_Path field specifying the routing information. These bytes indicate that the request is to be delivered out Port #2 and to MAC ID 9 and then out Port #4 to the Target Device at MAC ID 6.

#### **Step 2 – Request delivered to DeviceNet to DeviceNet Router**

When the EtherNet/IP to DeviceNet Router receives the Unconnected Send request with this Service Data field it detects that there are more intermediate devices to go through before the Target Device can be reached. In this case, the EtherNet/IP to DeviceNet Router executes the following steps:

- Executes the timing related logic associated with the *Priority/Tick Time* and *Connection Timeout Ticks* fields. If a timeout is detected, then a successful Unconnected Send response, which specifies a Routing Error, is returned to the Requesting Device.
- Strips its portion of the routing information from the Unconnected Send request's *Route\_Path* field and forwards the Unconnected Send Request accordingly. In this example, the "02 09" bytes constitute the routing information pertinent to DeviceNet Router #1 (next hop indicator). These bytes indicate that the Unconnected Send Request should be sent out Port #2 and to MAC ID #9 on that subnet.
- Establishes an Explicit Messaging Connection with the DeviceNet to DeviceNet Router (if necessary) and sends the remaining data in the Unconnected Send service the second router.

The Service Data field of the Unconnected Send Request that the EtherNet/IP to DeviceNet Router forwards to the DeviceNet to DeviceNet Router is presented below. In this example, the EtherNet/IP to DeviceNet Router is making use of the Transaction\_Id field for internal transaction management purposes and that the value "4" is being used.

**Table 10-2-10 Multiple Hop Unconnected Send Request Service Data (Second Hop)**

Bytes	Meaning
04 00	The Transaction_Id field inserted by the router.
07	Priority/Tick Time field (Time Tick = 128 ms).
08	Connection Timeout Ticks field (8 Ticks which results in 1024 ms timeout, 512 ms less than was originally received).
08 00	Message_Size field = 8 bytes.
0E	Service field = Get_Attribute_Single.
03	Request_Path_Size field indicates that there are 3 words in the request path.
20 01 24 02 30 03	Request Path field specifying 8 bit Class Id = 1, 8 bit Instance ID = 2, 8 bit Attribute Id = 3.
	Request_Data field empty
01	Route_Size field indicates that there is 1 word in the Route Path.
00	Reserved byte.
04 06	Route Path indicating that the request is to be delivered out Port #4 to the Target Device at MAC ID 6. Note that the EtherNet/IP to DeviceNet Router has stripped its routing information from the packet.

### **Step 3 – Request delivered to Remote Target Device**

When the DeviceNet to DeviceNet Router receives this Service Data field in the Unconnected Send Request it behaves identical to the DeviceNet Router in the previous step. Specifically, the DeviceNet to DeviceNet Router executes the following steps:

- Executes the timing related logic associated with the *Priority/Tick Time* and *Connection Timeout Ticks* fields. If a timeout is detected, then a successful Unconnected Send response that specifies a Routing Error is returned to the Requesting Device.
- Since there is no additional path routing, the device at MAC ID 6 is the Remote Target Device. The DeviceNet Router establishes an Explicit Messaging Connection with that device (if necessary), extracts the actual transaction from the Message Request portion of the Service Data, and delivers the Explicit Messaging Request to the Target Device.

The response returns back to the Requesting Device with each CIP Router (utilizing the Transaction\_Id field to facilitate internal transaction management when needed). If an error was detected at any point in the process, then the appropriate error information would be returned in the successful Unconnected Send Response.

## **10-3 Connected Explicit Message Routing to DeviceNet**

An explicit messaging connection is established using the Forward Open service of the Connection Manager. This section shall describe the mechanism for establishing an explicit messaging connection to a target on the subnet (the subnet is the last hop). Explicit Message routing through DeviceNet (the subnet is an intermediate hop) is described in section 10-5.

### **10-3.1 Explicit Message Connection Establishment to a DeviceNet Target**

Explicit message routing to DeviceNet can be from any CIP subnet to the DeviceNet subnet as the last (final) hop. A request for a connected Explicit Messaging connection is detected by the router when both of the following are true within the Forward Open service of the Connection Manager object:

- The DeviceNet subnet which the device is routing onto is the last hop (connection target is on this subnet)
- The Message Router class is specified in the connection path parameter

The DeviceNet router shall service the request by attempting to establish an Explicit Messaging connection to the target using either the UCMM or the Predefined Master/Slave Connection Set. This added complexity is necessary since the Forward Open service shall not be supported by a DeviceNet node when it is the target (final destination) of a connection request. The router shall attempt to acquire an explicit connection with the largest message body format for both class and instance.

### **10-3.2 Explicit Message Connection - Request**

The DeviceNet router shall process connected explicit messaging request data sent to a DeviceNet target device as described below:

1. Remove and save the Class 2/3 Sequence Count.
2. Save the Service Code for possible use during the response message.
3. Parse the Request Path field based on the Request Path Size field to extract the logical segments for the request.
4. If the Class or Instance value is larger than the message body format for the connection, discard the data.
5. Place the Service Code, Class, and Instance into the appropriate fields in the DeviceNet message. Attribute and Member ID (when applicable) are placed in the service data area, in that order.
6. The data contained in the Request\_Data field is appended to the end of the message.

### 10-3.3 Explicit Message Connection - Response

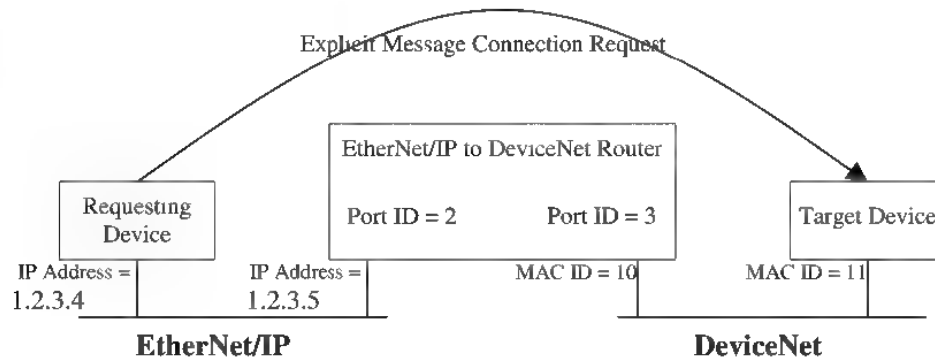
The DeviceNet router shall process connected explicit messaging response data sent from a DeviceNet target device as described below:

1. Insert the saved Class 2/3 Sequence Count maintained for each connection request and place it in the Sequence Count field for the connection data.
2. If the response indicated an error (0x94 response service code), place the saved request service code (with the high order bit set) into the Service Code field, the DeviceNet General Error code into the General Status field, a value of one into the Size of Additional Status field, and the DeviceNet Additional Code into the first UINT of the Additional Status field. The upper 8 bits of the Additional Status field entry shall be zero.
3. If the response did not indicate an error, place the service code value returned by the target device into the Service Code field, a value of zero into the General Status field, and a value of zero into the Size of Additional Status field.
4. Place any Service Data into the Response Data field.

### 10-3.4 Explicit Message Connection Establishment Example (Last Hop)

In the following example, a requesting device on a DeviceNet subnet establishes an Explicit Messaging connection to a target device on a different DeviceNet subnet.

**Figure 10-3-1 One-hop Connection Request**



The table below presents the contents of the Service Data field for the Forward Open request, which is used to execute the connection establishment example noted above (all values are in hex).

**Table 10-3-1 Forward Open Request Service Data**

Byte Value	Meaning
pp	Priority/Tick Time field.
tt	Connection Timeout Ticks field.
ww xx yy zz	O_to_T Connection ID – Chosen by originating node
FF FF FF FF	T_to_O Connection ID – Chosen by target node
0C 00	Connection Serial Number
FF FF	Originator Vendor ID (Vendor ID = 65535)
00 01 02 03	Originator Serial Number (Serial Number = 0x03020100)
00	Connection Timeout Multiplier
00 00 00	Reserved
80 96 98 00	O to T RPI (0x989680 = 10 s)
FF 42	O_to_T Connection Parameters
80 96 98 00	T to O RPI (0x989680 = 10 s)
FF 42	T_to_O Connection Parameters
83	Transport Type / Trigger (Class 2 Server)
05	Connection_Path_Size field indicates that there are 5 words in the connection path
03 0B 21 00 02 00 25 00 01 00	Contents of the Connection_Path field. This field specifies the routing/connection information. These bytes indicate that the request is to be delivered out Port #3 and to MAC ID 11. It further indicates the application being connected to by specifying 16 bit Class ID = 2 (Message Router), 16 bit Instance ID = 1.

**Step 1 – Request delivered to the EtherNet/IP to DeviceNet Router**

The first step in the process calls for the Requesting Device to issue the Forward Open request across the EtherNet/IP UCMM.

**Step 2 – DeviceNet Router internally routes message to DeviceNet port**

The EtherNet/IP port detects that this message is not intended for itself and delivers the request to the Connection Manager on the DeviceNet port.

### **Step 3 – DeviceNet Router creates connection with the Target Device**

After the Connection Manager Object within the DeviceNet Router processes the Forward Open request it examines the contents of the Connection\_Path field. In this case, the contents are “03 0A 21 00 02 00 25 00 01 00”. This indicates that the next device in the hop is on Port #3 and its Node Address (MAC ID) is 11. Additionally, since there is only a single Port/Node Address pair specified in the Connection\_Path field, the request has reached its destination network. When a DeviceNet Router receives a Forward Open Request that does not need to be forwarded through more intermediate CIP Routers, and the Message Router is specified as the application object, the following steps are taken:

- The DeviceNet Router executes the timing related logic associated with the Priority/Tick Time and Connection Timeout Ticks fields. If a timeout is detected, then a successful Forward Open response that specifies a Routing Error is returned to the Requesting Device.
- If an explicit messaging connection to the target is not already open, the DeviceNet Router creates an Explicit Message connection with the Target Device using the UCMM (or the Predefined Connection Set mechanism if the UCMM attempts fail).
- The expected\_packet\_rate attribute of the target connection object is set to the value requested in the Forward Open service.

### **Step 4 – DeviceNet Router processes Open Response from Target Device**

If the connection request was successful, the DeviceNet router completes the connection establishment internally and returns a successful response. If it fails (target node does not exist, is out of resources, etc.) then the DeviceNet router shall return the appropriate error response.

## **10-4 I/O Message Routing to DeviceNet**

I/O message routing to DeviceNet can be from any CIP subnet to the DeviceNet subnet as the last (final) hop. When the DeviceNet subnet which the device is routing onto is the last hop (connection target is on this subnet) within the Forward Open service of the Connection Manager object, then the routing device shall establish the connection(s) to the target using either dynamic I/O connections or the Predefined Master/Slave Connection Set. This added complexity is necessary since the Forward Open service shall not be supported by a DeviceNet node when it is the target (final destination) of a connection request.

### **10-4.1 I/O Connection Establishment to a DeviceNet Target**

When the DeviceNet subnet which the device is routing onto is the last hop (connection target is on this subnet), then the routing device shall perform the following steps to establish the I/O connection.

1. Verify connection parameters
2. If an Electronic Key segment is present, verify key values at target.
3. If the T\_to\_O Network Connection Parameter indicates Multicast and this is a transport class 0/1 request, check for existence of a connection producing this data. If found, use that connection instance for target production and go to Step 5.
4. Attempt to create a dynamic I/O connection to the target device. If the dynamic I/O connections are not supported go to Step 11.



5. If the Forward Open has two transport class 0/1 I/O connections specified, create a second dynamic I/O connection and bind the two newly created connections.
6. Map the Forward Open connection parameters to individual Set Attribute Single message requests to the target device.
7. Connection ID selection.
8. Process the Production Inhibit Time Network Segment, if applicable.
9. If the Forward Open specified a Configuration path and a Data Segment is present, send the data in the Data Segment to the target device using the Configuration path specified.
10. Send ApplyAttributes service to the connection instance(s). Process response from target per specification of the Connection Object within the CIP Common Specification. Go to Step 17.
11. Attempt Predefined Master/Slave Connection Set allocation.
12. Verify that the specified Connection Timeout Multiplier can be supported.
13. Compare Consumed and Produced connection paths to those stored in the target nodes connection instances, where applicable. If the paths do not match, attempt to set the paths using the Set Attribute Single service. If the set fails, return error.
14. Process the Production Inhibit Time Network Segment, if applicable.
15. If the Forward Open specified a Configuration path and a Data Segment is present, send the data in the Data Segment as a Set Attribute Single message request to the target device using the Configuration path specified.
16. Send Set Attribute Single service to the expected\_packet\_rate attribute of the connection instance(s).
17. Return success response to the Forward Open service.

If any error occurs during the establishment of a dynamic I/O connection, the router shall attempt the Predefined Connection Set (transition to Step 11).

For additional routing and bridging requirements specific to safety implementations see CIP Safety Specification (Volume 5, Chapter 10).

#### **10-4.1.1 Verify Connection Parameters**

The router shall verify that it can process the values of the parameters in the Forward Open request and that they are valid for targets on the DeviceNet subnet.

#### **10-4.1.2 Electronic Key Verification**

When an Electronic Key segment is present for the last hop, the router shall be responsible for electronic key verification at the target. DeviceNet does not provide for delivery of the key, so the router shall issue Set Attribute Single requests to Instance 1 of the Identity object within the target for each key. The Attribute IDs placed in the request depend on the key format parameter in the Electronic Key segment. The router shall compare the value returned from the target with the value in the key segment for each key parameter (ie: Vendor ID, Product Code, etc.).

For an Electronic Key Format Value 4, if the value in the key field is zero then the router does not need to get the value at the target and the key comparison is successful for that field. If the compatibility bit is set, all fields except the major and minor revision fields shall match; the comparison of the revision fields is successful if the major.minor revision of the target is greater than or equal to that of the major and minor revision key fields.

#### **10-4.1.3 Multicast Production at Target Check**

If the T\_to\_O Network Connection Parameter Connection Type specifies Multicast, and this is a transport class 0/1 connection, the router shall check for a connection already producing the data on the target device. To accomplish this, the router can send the Multicast Lookup service to the target using the producing application object path specified in the Forward Open. If a success is returned, the router shall use the resources of that connection on the target for the target data production. If it is not successful, the router shall configure the connection(s) as though they are point to point.

#### **10-4.1.4 Dynamic I/O Connection Creation Attempt**

A router attempts to create a dynamic I/O connection by sending a Create service to the Connection Class at the target device.

#### **10-4.1.5 Second Dynamic I/O Connection Creation, if Required**

If the Forward Open has two transport class 0/1 I/O connections specified, the router shall send a second Create service. If successful, the two I/O connections are 'bound' together by sending the Connection Bind service to the Connection Class with the instance numbers of this connection instance and the previously created connection instance as parameters. If the Connection Bind service fails, return error. The router shall designate one connection instance as the client and the other as the server.

#### **10-4.1.6 Mapping Forward Open Connection Parameters**

The DeviceNet router maps and/or processes the Forward Open service parameters as described in this section.

##### **10-4.1.6.1 Priority/Time\_tick and Time-out\_ticks**

The *Priority/Time\_tick* and *Time-out\_ticks* parameters are processed as specified by the Connection Manager object. The CIP router shall attempt to determine, as best it can, if the request can be completed in the remaining amount of time. However, it does not need to subtract the value for this time, nor deliver these parameters to the target node.

##### **10-4.1.6.2 O\_to\_T and T\_to\_O Network Connection IDs**

These connection IDs are used for the subnet, which the request came in on; thus, they are irrelevant to the target DeviceNet subnet and are handled as specified by the Connection Manager object.

#### **10-4.1.6.3 Connection Serial Number, Originator Vendor ID, and Originator Serial Number**

These parameters are not passed along to the target node.

#### **10-4.1.6.4 Connection Timeout Multiplier**

If this parameter specifies a timeout multiplier of 4 then no action needs to be taken. If this parameter specifies a multiplier other than 4, send a Set Attribute Single service to the connection\_timeout\_multiplier attribute of each connection instance using the value from the parameter. If the target returns an error response, then this Forward Open request cannot be handled and an error shall be returned to the originator.

#### **10-4.1.6.5 O\_to\_T and T\_to\_O RPI**

If two class 0/1 instances have been instantiated, the RPI values are sent to the respective expected\_packet\_rate attributes of the producing and consuming connection instances. If a single class 2/3 instance has been instantiated, the O\_to\_T RPI value is sent if the target is the server while the T\_to\_O RPI value is sent if the target is the client.

The RPI values are converted from microseconds to milliseconds before sending to the target via a Set Attribute Single service. The converted values shall be rounded to the next highest millisecond. The value returned from the target in response to the Set Attribute Single service request is retained, converted from milliseconds to microseconds, and used in the successful Forward Open response message.

#### **10-4.1.6.6 O\_to\_T and T\_to\_O Network Connection Parameters**

The Network Connection Parameters are mapped as follows:

- **Connection Size:** The Connection Size value(s) are sent to the appropriate produced and consumed attribute connection size attribute(s).
- **Fixed/Variable:** This field is ignored.
- **Priority:** The router shall interpret the Priority field as follows:
  - Low - Use Group 3 identifiers
  - Medium - Use Group 2 identifiers
  - High - Use Group 1 identifiers
  - Urgent - Not supported; return error
- **Connection Type:**
  - Null – Not supported, return error
  - Multicast – Attempt to use existing connection instances
  - Point to Point – Create new connection instances
- **Redundant Owner:** If this bit is set (1) then the Forward Open request shall be rejected.

#### **10-4.1.6.7 Transport Type/Trigger**

If a single connection has been created, the Transport Type/Trigger value is sent to the transportClass\_trigger attribute within the target connection object without modification. If two Class 0/1 connections have been created, send the trigger and transport class values along with a Client direction to the designated client connection instance and the transport class value along with a Server direction to the designated server connection instance.

#### **10-4.1.6.8 Connection\_Path**

The consumed and/or produced connection paths are parsed from the connection path parameter.

#### **10-4.1.7 Connection ID Selection**

Connection ID selection is performed as specified by the Connection Object in Chapter 3 of this specification.

#### **10-4.1.8 Handling Configuration Data in Forward Open**

If the Forward Open specifies a Configuration path and a Data Segment is present, the data in the Data Segment is sent as a Set Attribute Single message request to the target device using the Configuration path specified.

#### **10-4.1.9 Process the Production Inhibit Time Network Segment**

If the Forward Open connection path indicated a Production Inhibit Time Network Segment for the DeviceNet subnet, send the specified production inhibit time value to the production\_inhibit\_time attribute of the producing connection object on the target device.

#### **10-4.1.10 Establishing the Dynamic I/O Connection with the ApplyAttributes Service**

Once all connection object attributes are successfully set based on the Forward Open parameters, each I/O connection is transitioned to the Established state by sending the ApplyAttributes service request. A success response may contain connection IDs allocated by the target. A failure results in an error response returned in the Forward Open response message.

#### 10-4.1.11 Predefined Master/Slave Connection Set allocation

The selection of the Allocation Choice is based on the Forward Open parameters as shown in the following table. The Trigger Type parameter is always valid (even when the direction bit is set to Server) in order to allow the DeviceNet router to determine the correct Allocation Choice. In all cases except the Application Triggered / Class 2/3 request a Connection Type of Multicast is invalid and an error is returned.

**Table 10-4-1 Allocation Choice Selections**

Trigger Type	Transport Class		
	One Class 0 (O to T Connection Type Null)	One Class 0 (T to O Connection Type Null)	Two Class 0 or One Class 2/3
Cyclic	Invalid, return error	Allocate Cyclic (instance 4 only) with Ack Suppresion	Attempt Cyclic Allocation with Ack Suppresion If Cyclic fails, attempt Poll Allocation <sup>1</sup>
Change of State	Invalid, return error	Allocate COS (instance 4 only) with Ack Suppression	Allocate COS with Ack Suppression
Application Triggered	Invalid, return error	Invalid, return error	Attempt Poll <sup>1</sup> or Strobe <sup>2,3</sup> Allocation <sup>4</sup>

<sup>1</sup> The router shall configure the 'Poll' connection to the rate specified in the Forward Open and produce data to the target device when the router receives it from the client.

<sup>2</sup> The router shall configure the 'Strobe' connection to the rate configured within the router for Strobe/Poll devices and shall produce the most recent data based on the 'scan cycle'. Thus, the expected packet rate sent in the Forward Open is not used. The router may attempt to verify that the RPI can be met and if it cannot, return an error during connection establishment.

<sup>3</sup> If Strobe is selected by the originator, the O to T connection size shall be 1 byte (of which only the low order bit is used), and the T to O connection size shall not be greater than 8.

<sup>4</sup> The router shall attempt to allocate the Poll connection if the O to T Connection Type is 'Point to Point or the Strobe connection if the O to T Connection Type is Multicast.

#### 10-4.1.12 Verification of Connection Timeout Multiplier

If the Connection Timeout Multiplier parameter specifies a timeout multiplier of 4 then no action needs to be taken. If this parameter specifies a multiplier other than 4, send a Set Attribute Single service to the connection\_timeout\_multiplier attribute of each connection instance. If the target returns an error response then this Forward Open request can not be handled and an error returned to the originator.

#### 10-4.1.13 Verification of Predefined Connection Paths

Compare Consumed and Produced connection paths to those stored in the target nodes connection instances, where applicable. If the paths do not match, attempt to set the paths using the Set Attribute Single service. If the set fails, return error.

#### 10-4.1.14 Handling Configuration Data to a Group 2 Server

The router, as a Group 2 Client, handles configuration data the same when establishing a dynamic I/O connection. See Chapter 10-4.1.8.

#### 10-4.1.15 Establishing the Predefined I/O Connection by Setting the EPR Attribute

Each I/O connection is transitioned to the Established state by sending a SetAttributeSingle service request to the expected\_packet\_rate attribute in the Group 2 server's I/O connection instance. See Chapter 3 for more details.

The value returned from the target in a successful SetAttributeSingle response is the actual expected packet rate used at the target. If an error is returned, the 'Apply\_Attributes' portion of the request failed.

#### 10-4.1.16 Returning a Success Response

The DeviceNet router shall load the appropriate values into the Forward Open response message and return success.

### 10-4.2 I/O Connection Usage

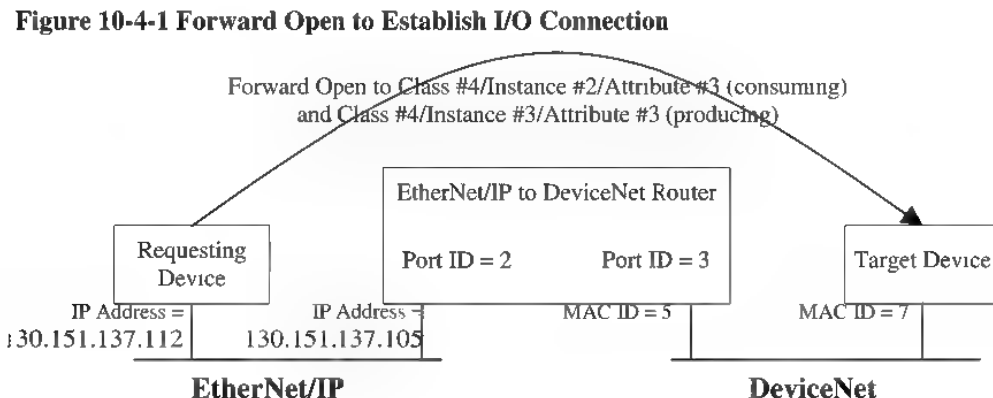
When a Class 0/1 transport class is used, pass the data on.

When a Class 2/3 transport class is used, the DeviceNet router shall:

- Remove and discard the sequence count from the connected I/O data sent to the DeviceNet target.
- Maintain a sequence count for each connection.
- Increment and load into the connection data the sequence count for each message sent from the DeviceNet target.

### 10-4.3 I/O Connection Establishment Example

In the following example, a requesting device on an EtherNet/IP subnet establishes an I/O connection to a target device on a DeviceNet subnet. Assume that the Requesting Device wants to make the I/O connection to both a consuming and producing I/O Assembly. The address in the Target Device for the consuming Assembly is identified as Class #4/Instance #2/Attribute ID #3. The producing Assembly is identified as Class #4/Instance #3/Attribute #3. The request is for a Point to Point I/O connection which consumes 4 bytes and produces 6 bytes.



The table below presents the contents of the Service Data field for the Forward Open request that is used to execute the connection establishment example noted above (all values are in hex).

**Table 10-4-2 Forward Open Request Service Data**

Byte Value	Meaning
pp	Priority/Tick Time field.
tt	Connection Timeout Ticks field
ww xx yy zz	O to T Connection ID Chosen by originating node
FF FF FF FF	T_to_O Connection ID – Chosen by target node
0C 00	Connection Serial Number
FF FF	Originator Vendor ID (Vendor ID = 65535)
00 01 02 03	Originator Serial Number (Serial Number = 0x03020100)
00	Connection Timeout Multiplier
00 00 00	Reserved
20 A1 07 00	O_to_T RPI (0x7A120 = 500 ms)
04 48	O_to_T Connection Parameters
20 A1 07 00	T_to_O RPI (0x7A120 = 500 ms)
06 48	T to O Connection Parameters
82	Transport Type / Trigger (Class 2 Cyclic Server)
09	Connection_Path_Size field indicates that there are 9 words in the connection path.
03 07 21 00	Contents of the Connection_Path field. This field specifies the routing/connection information. These bytes indicate that the request is to be delivered out Port #3 and to MACID 7. It further indicates the applications being connected to by specifying 16 bit Class ID = 4, 16 bit Instance ID = 2, 8 bit Attribute ID = 3 (consuming I/O path) and 16 bit Class ID = 4, 16 bit Instance ID = 3, 8 bit Attribute ID = 3 (producing I/O path).
04 00 25 00	
02 00 30 03	
21 00 04 00	
25 00 03 00	
30 03	

#### **Step 1 – Request delivered to the EtherNet/IP to DeviceNet Router**

The first step in the process calls for the Requesting Device to issue the Forward Open request to the EtherNet/IP to DeviceNet router. This originating device uses the UCMM on EtherNet/IP.

#### **Step 2 – Router creates a CIP Bridged connection**

Upon receipt of a Forward Open to the Connection Manager object, the router shall (if it has the resources available) create a CIP Bridged connection (Class Code 0x05) and place it in the Configuring state. The remaining attributes are set with either the values received in the service request or the appropriate default values. If resources are unavailable, an error response is returned.

#### **Step 3 – Router creates connection with the Target Device**

After the Connection Manager Object within the DeviceNet Router processes the Forward Open request it examines the contents of the Connection\_Path field. In this case, the contents are “03 07 21 00 04 00 25 00 02 00 30 03 00”. This indicates that the next device in the hop is on Port #3 and its Node Address (MACID) is 7. Additionally, since there is only a single Port/Node Address pair specified in the Connection\_Path field, the request has reached its destination network. When a DeviceNet Router receives an Forward Open Request that DOES

NOT need to be forwarded through more intermediate CIP Routers, the following steps are taken:

- The DeviceNet Router executes the timing related logic associated with the Priority/Tick Time and Connection Timeout Ticks fields. If a timeout is detected, then a successful Forward Open response that specifies a Routing Error is returned to the Requesting Device.
- The DeviceNet Router creates an I/O connection with the Target Device. Once the I/O connection is created, the connection attributes are set, and the connection is applied.

In the example above, the connection path inside the Forward Open indicates an I/O connection to Class #4, Instance #2, Attribute #3.

#### **Step 4 – Router returns the Forward Open Response to the Requesting Device**

The DeviceNet Router then delivers the Forward Open Response to the original Requesting Device. This response will contain the actual packet rates and the connection identifiers to be used.

The service data field for a Forward Open response is shown below.

**Table 10-4-3 Forward Open Response Service Data**

Byte Value	Meaning
45 00 00 00	O_to_T Connection ID – Chosen by originating node (CAN ID 0x045 = Node 5, Group 1, Msg ID 1)
07 01 00 00	T to O Connection ID – Chosen by target node (CAN ID 0x107 = Node 7, Group 1, Msg ID 4)
0C 00	Connection Serial Number
FF FF	Originator Vendor ID (Vendor ID = 65535)
00 01 02 03	Originator Serial Number (Serial Number = 0x03020100)
00	Connection Timeout Multiplier
00 00 00	Reserved
20 A1 07 00	O to T API (0x7A120 = 500 ms)
20 A1 07 00	T_to_O API (0x7A120 = 500 ms)
00	Application reply size (no application reply data)
00	Reserved

Assume that the connection to the Target Device was successful. The text below presents the contents of the CAN Data Field associated with the Forward Open Response returned to the Requesting Device:



**Table 10-4-4 Forward Open Response on DeviceNet**

Bytes	Meaning
07	Frag = 0, Transaction ID = 0, Destination MAC = 7.
D4	Service = D4. In this context, this specifies a response to the Forward Open service.
00	The General Status field. The value zero (0) indicates that there were no routing errors.
00	Reserved byte.
45 00 00 00 07 01 00 00 0C 00 FF FF 00 01 02 03 00 00 00 00 20 A1 07 00 20 A1 07 00 00 00	Forward Open Service Response Data. This is the previously described Forward Open service data described above.

## 10-5 Connected Message Routing Through/From DeviceNet

The DeviceNet router does not need to distinguish between Explicit Message and I/O connected Message routing when acting as an intermediate hop, or when receiving a request from DeviceNet. In both cases a CIP Bridged connection is established.

For additional routing and bridging requirements specific to safety implementations see CIP Safety Specification (Volume 5, Chapter 10).

### 10-5.1 I/O and Explicit Message Routing Through DeviceNet (Intermediate Hop)

When acting as a router on an intermediate hop (the request is directed at the DeviceNet subnet, but for a target not on a different subnet), the router will process the Forward Open as defined in the CIP Common specification. The local target is another DeviceNet router, and the Forward Open is propagated across the subnet. The connected data is not processed for explicit messaging, and sequence counts are not removed/inserted.

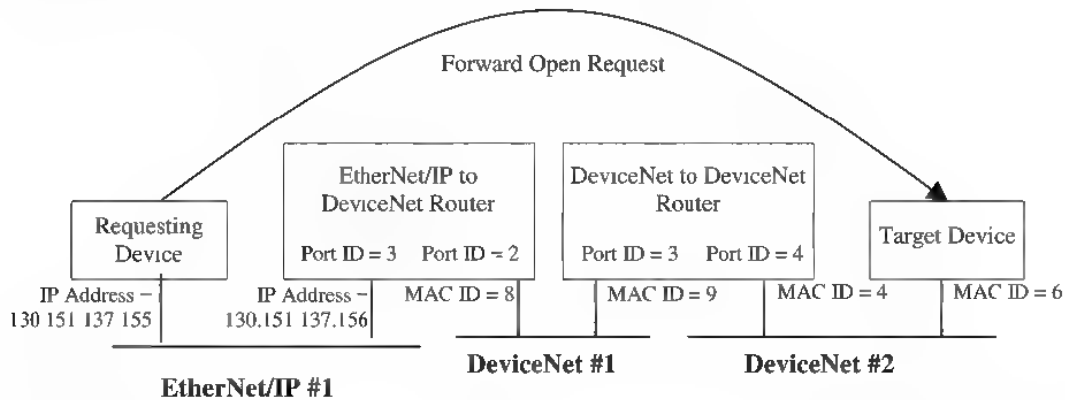
### 10-5.2 I/O and Explicit Message Routing From DeviceNet

When acting as a router for an originator on DeviceNet (the originating device is on the subnet and the target is on a different subnet), the same conditions apply as described in the intermediate hop section above. The router cannot/does not distinguish between the two cases.

### 10-5.3 I/O Message Routing Through DeviceNet Example (Intermediate Hop)

Assume that the Requesting Device wants to make the I/O connection to both a consuming and producing I/O Assembly. The address in the Target Device for the consuming Assembly is identified as Class #4/Instance #2/Attribute ID #3. The producing Assembly is identified as Class #4/Instance #3/Attribute #3. The request is for a Point to Point I/O connection which consumes 4 bytes and produces 6 bytes.

Figure 10-5-1 Forward Open to Establish I/O Connection



The table below presents the contents of the Service Data field for the Forward Open request that is used to execute the connection establishment example noted above (all values are in hex).

Table 10-5-1 Forward Open Request Service Data

Byte Value	Meaning
pp	Priority/Tick Time field
tt	Connection Timeout Ticks field.
ww xx yy zz	O_to_T Connection ID – Chosen by originating node
FF FF FF FF	T_to_O Connection ID Chosen by target node
0C 00	Connection Serial Number
FF FF	Originator Vendor ID (Vendor ID = 65535)
00 01 02 03	Originator Serial Number (Serial Number = 0x03020100)
00	Connection Timeout Multiplier
00 00 00	Reserved
20 A1 07 00	O_to_T RPI (0x7A120 = 500 ms)
04 48	O to T Connection Parameters
20 A1 07 00	T_to_O RPI (0x7A120 = 500 ms)
06 48	T to O Connection Parameters
82	Transport Type / Trigger (Class 2 Cyclic Server)
0A	Connection_Path_Size field indicates that there are 10 words in the connection path.
02 09 04 06 21 00 04 00 25 00 02 00 30 03 21 00 04 00 25 00 03 00 30 03	Contents of the Connection_Path field. This field specifies the routing/connection information. These bytes indicate that the request is to be delivered out Port #2 and to Node Address 9, then out Port #4 and to Node Address 6. It further indicates the application being connected to by specifying 16 bit Class ID = 4, 16 bit Instance ID = 2, 8 bit Attribute ID = 3.

#### Step 1 – Request delivered to the EtherNet/IP to DeviceNet Router

The first step in the process calls for the Requesting Device to issue the Forward Open request to the EtherNet/IP to DeviceNet Router. When received at the EtherNet/IP port of the router it is processed and sent to the Connection Manager on the DeviceNet port based on the contents of the Connection Path field.

### **Step 2 – Router creates a CIP Bridged connection**

Upon receipt of the Forward Open by the Connection Manager on the DeviceNet port, the router (Node Address 8) shall (if it has the resources available) create a CIP Bridged connection (Class Code 0x05) internally and place it in the Configuring state. The remaining attributes are set with either the values received in the service request or the appropriate default values. If resources are unavailable, an error response is returned.

### **Step 3 – Router sends request to next router**

The EtherNet/IP to DeviceNet router sends the Forward Open request to the DeviceNet to DeviceNet router after removing its path from the Connection\_Path parameter. It also chooses a Connection ID to use. The table below shows the data for this request where the EtherNet/IP to DeviceNet router selected Group 1, Message ID 4 (108<sub>hex</sub>).

**Table 10-5-2 Forward Open Request on DeviceNet**

Bytes	Meaning
09	Frag = 0 <sup>1</sup> , Transaction ID = 0, Destination MACID = 9.
54	Service = 54. In this context, this specifies the Forward Open service
06	Class ID of the Connection Manager Object Class.
01	The Instance ID Field specifying instance 1 of the Connection Manager.
pp tt 08 01 00 00 FF FF FF FF 0C 00 FF FF 00 01 02 03 03 00 00 00 00 20 A1 07 00 31 35 31 2E 31 33 37 2E 31 30 35 00	Forward Open Service Request Data. This is the Forward Open service data described above.

<sup>1</sup> - This request would have to be fragmented on DeviceNet to deliver it to the router. DeviceNet fragmentation is not illustrated in this example.

The Forward Open request has now been delivered to the DeviceNet Router via an Explicit Messaging Connection that exists between the Originating Device and the DeviceNet Router.

### **Step 3 – Router creates connection with the Target Device**

After the Connection Manager Object within the DeviceNet Router processes the Forward Open request it examines the contents of the Connection\_Path field. In this case, the contents are “13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 21 00 04 00 25 00 02 00 30 03 00”. This indicates that the next device in the hop is on Port #3 and its Node (IP) Address is 130.151.137.105. Additionally, since there is only a single Port/Node Address pair specified in the Connection Path field, the request has reached its destination network. When a DeviceNet Router receives a Forward Open Request that DOES NOT need to be forwarded through more intermediate CIP Routers, the following steps are taken:

- The DeviceNet Router executes the timing related logic associated with the Priority/Tick Time and Connection Timeout Ticks fields. If a timeout is detected, then a successful Forward Open response that specifies a Routing Error is returned to the Requesting Device.
- The DeviceNet Router creates an I/O connection with the Target Device. Once the I/O connection is created, the connection attributes are set, and the connection is applied. (Note that if the connection path indicates the Message Router object, then the router opens an Explicit Messaging connection using the UCMM.)

In the example above, the connection path inside the Forward Open indicates an I/O connection to Class #4, Instance #2, Attribute #3.

#### **Step 4 – Router returns the Forward Open Response to the Requesting Device**

The DeviceNet Router then delivers the Forward Open Response to the original Requesting Device. This response will contain the actual packet rates and the connection identifiers to be used.

The service data field for a Forward Open response is shown below.

**Table 10-5-3 Forward Open Response Service Data**

Byte Value	Meaning
45 00 00 00	O_to_T Connection ID – Chosen by originating node (CAN ID 0x045 = Node 5, Group 1, Msg ID 1)
07 01 00 00	T to O Connection ID Chosen by target node (CAN ID 0x107 = Node 7, Group 1, Msg ID 4)
0C 00	Connection Serial Number
FF FF	Originator Vendor ID (Vendor ID = 65535)
00 01 02 03	Originator Serial Number (Serial Number = 0x03020100)
00	Connection Timeout Multiplier
00 00 00	Reserved
20 A1 07 00	O to T API (0x7A120 = 500 ms)
20 A1 07 00	T_to_O API (0x7A120 = 500 ms)
00	Application reply size (no application reply data)
00	Reserved

Assume that the connection to the Target Device was successful. The text below presents the contents of the CAN Data Field associated with the Forward Open Response returned to the Requesting Device:

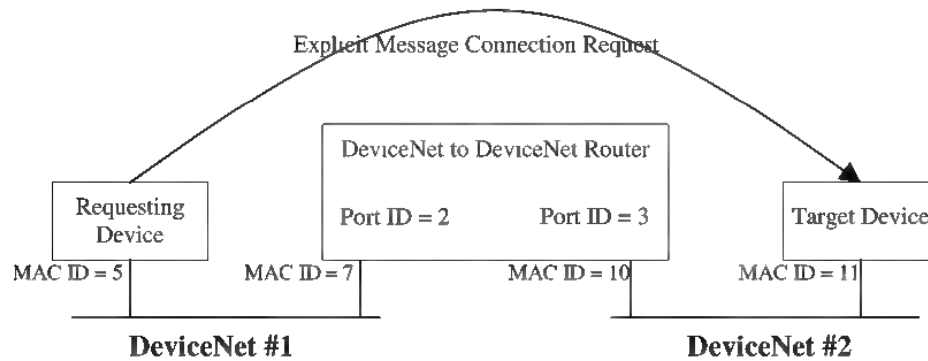
**Table 10-5-4 Forward Open Response on DeviceNet**

Bytes	Meaning
07	Frag = 0, Transaction ID = 0, Destination MAC = 7.
D4	Service = D4. In this context, this specifies a response to the Forward Open service.
00	The General Status field. The value zero (0) indicates that there were no routing errors.
00	Reserved byte.
45 00 00 00 07 01 00 00 0C 00 FF FF 00 01 02 03 00 00 00 00 20 A1 07 00 20 A1 07 00 00 00	Forward Open Service Response Data. This is the Forward Open service data described above.

### **10-5.4 Explicit Message Routing from DeviceNet Example**

In the following example, a requesting device on a DeviceNet subnet establishes an Explicit Messaging connection to a target device on a different DeviceNet subnet.

Figure 10-5-2 One-hop Explicit Connection Request



The table below presents the contents of the Service Data field for the Forward Open request which is used to execute the connection establishment example noted above (all values are in hex).

Table 10-5-5 Forward Open Request Service Data

Byte Value	Meaning
pp	Priority/Tick Time field.
tt	Connection Timeout Ticks field
45 00 00 00	O to T Connection ID Chosen by originating node (CAN ID 0x045 = Node 5, Group 1, Msg ID 1)
FF FF FF FF	T_to_O Connection ID – Chosen by target node
0C 00	Connection Serial Number
FF FF	Originator Vendor ID (Vendor ID = 65535)
00 01 02 03	Originator Serial Number (Serial Number = 0x03020100)
00	Connection Timeout Multiplier
00 00 00	Reserved
80 96 98 00	O_to_T RPI (0x989680 = 10 s)
	O_to_T Connection Parameters
80 96 98 00	T_to_O RPI (0x989680 = 10 s)
	T_to_O Connection Parameters
83	Transport Type / Trigger (Class 2 Server)
05	Connection_Path_Size field indicates that there are 5 words in the connection path.
03 0B 21 00 02 00 25 00 01 00	Contents of the Connection_Path field. This field specifies the routing/connection information. These bytes indicate that the request is to be delivered out Port #3 and to MAC ID 11. It further indicates the application being connected to by specifying 16 bit Class ID = 2 (Message Router), 16 bit Instance ID = 1

#### Step 1 – Request delivered to the DeviceNet to DeviceNet Router

The first step in the process calls for the Requesting Device to establish an Explicit Messaging Connection with the DeviceNet to DeviceNet Router and issue the Forward Open request across that connection. The table below presents the entire Forward Open request. Assume that the Message Body Format established for the Explicit Messaging Connection is DeviceNet 8/8.

**Table 10-5-6 Forward Open Request on DeviceNet**

Bytes	Meaning
07	Frag = 0 <sup>1</sup> , Transaction ID = 0, Destination MACID = 7.
54	Service = 54. In this context, this specifies the Forward Open service.
06	Class ID of the Connection Manager Object Class.
01	The Instance ID Field specifying instance 1 of the Connection Manager.
pp tt 45 00 00 00 FF FF FF 0C 00 FF FF 00 01 02 03 00 00 00 00 80 96 98 00	Forward Open Service Request Data. This is the Forward Open service data described above.

<sup>1</sup> - This request would have to be fragmented on DeviceNet to deliver it to the router. DeviceNet fragmentation is not illustrated in this example.

The Forward Open request has now been delivered to the DeviceNet Router via an Explicit Messaging Connection that exists between the Originating Device and the DeviceNet Router.

### **Step 2 – Router creates a CIP Bridged connection**

Upon receipt of a Forward Open to the Connection Manager object, the router shall (if it has the resources available) create a CIP Bridged connection (Class Code 0x05) and place it in the Configuring state. The remaining attributes are set with either the values received in the service request or the appropriate default values. If resources are unavailable, an error response is returned.

### **Step 3 – Router creates connection with the Target Device**

After the Connection Manager Object within the DeviceNet Router processes the Forward Open request it examines the contents of the Connection\_Path field. In this case, the contents are “03 0A 21 00 02 00 25 00 01 00”. This indicates that the next device in the hop is on Port #3 and its Node Address (MAC ID) is 11. Additionally, since there is only a single Port/Node Address pair specified in the Connection\_Path field, the request has reached its destination network. When a DeviceNet Router receives a Forward Open Request that does not need to be forwarded through more intermediate CIP Routers, and the Message Router is specified as the application object, the following steps are taken:

- The DeviceNet Router executes the timing related logic associated with the Priority/Tick Time and Connection Timeout Ticks fields. If a timeout is detected, then a successful Forward Open response that specifies a Routing Error is returned to the Requesting Device.
- The DeviceNet Router creates an Explicit Message connection with the Target Device using the UCMM.
- The expected\_packet\_rate attribute of the target connection object is set to the value requested in the Forward Open service.

## 10-6 Closing Connections

### 10-6.1 Using the Forward Close Service to Close an I/O Messaging Connection

The table below presents the contents of the Service Data field for the Forward Close request to delete the connection, which was established in the example above (all values are in hex).

**Table 10-6-1 Forward Close Request Service Data**

Byte Value	Meaning
pp	Priority/Tick Time field.
tt	Connection Timeout Ticks field.
0C 00	Connection Serial Number
FF FF	Originator Vendor ID (Vendor ID = 65535)
00 01 02 03	Originator Serial Number (Serial Number = 0x03020100)
0E	Connection_Path Size field indicates that there are 14 words in the connection path.
00	Reserved
13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 21 00 04 00 25 00 02 00 30 03 00	Contents of the Connection_Path field. This field specifies the routing/connection information. These bytes indicate that the request is to be delivered out Port #3 and to IP Address 130.151.137 105. This path encoding uses the Extended Link Address format for the IP address. It further indicates the application being disconnected from by specifying 16 bit Class ID = 4, 16 bit Instance ID = 2, 8 bit Attribute ID = 3

#### **Step 1 – Request delivered to the DeviceNet to EtherNet/IP Router**

The first step in the process calls for the Requesting Device to establish an Explicit Messaging Connection with the DeviceNet to EtherNet/IP Router (if one does not exist) and issue the Forward Close request. The table below presents the entire Forward Open request. Assume that the Message Body Format established for the Explicit Messaging Connection is DeviceNet 8/8.

**Table 10-6-2 Forward Close Request on DeviceNet**

Bytes	Meaning
07	Frag = 0 <sup>1</sup> , Transaction ID = 0, Destination MACID = 7.
5E	Service = 5E In this context, this specifies the Forward Close service.
06	Class ID of the Connection Manager Object Class.
01	The Instance ID Field specifying instance 1 of the Connection Manager.
01 00 pp tt 0C 00 0E 05 21 00 04 00 25 00 02 00 30 03 09 00 13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00	Forward Close Service Request Data. This is the Forward Close service data described above.

<sup>1</sup> - This request would have to be fragmented to deliver it to the router. DeviceNet fragmentation is not illustrated in this example.

The Forward Close request has now been delivered to the DeviceNet Router via an Explicit Messaging Connection that exists between the Originating Device and the DeviceNet Router.

### **Step 2 – Router deletes the CIP Bridged connection with Target**

After the Connection Manager Object within the DeviceNet Router processes the Forward Close request it examines the contents of the *Connection\_Path* field. In this case, the contents are “13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 21 00 04 00 25 00 02 00 30 03 00”. This indicates that the next device in the hop is on Port #3 and its Node (IP) Address is 130.151.137.105. Additionally, since there is only a single Port/Node Address pair specified in the *Connection\_Path*, the request has reached its destination network. When a DeviceNet Router receives an Forward Close Request that DOES NOT need to be forwarded through more intermediate CIP Routers, the following steps are taken:

The DeviceNet Router executes the timing related logic associated with the *Priority/Tick Time* and *Connection Timeout Ticks* fields. If a timeout is detected, then a successful Forward Close response that specifies a Routing Error is returned to the Requesting Device.

The DeviceNet Router deletes the I/O or Explicit Messaging connection with the Target Device.

### **Step 3 – Router returns the Forward Close Response to the Requesting Device and releases internal resources**

The DeviceNet Router then delivers the Forward Close Response to the original Requesting Device and releases all internal resources related to the connections on both the DeviceNet and Ethernet/IP subnets.

The service data field for a Forward Close response is shown below.

**Table 10-6-3 Forward Close Response Service Data**

Byte Value	Meaning
0C 00	Connection Serial Number
FF FF	Originator Vendor ID (Vendor ID = 65535)
00 01 02 03	Originator Serial Number (Serial Number = 0x03020100)
00	Application reply size (no application reply data)
00	Reserved

Assume that the connection to the Target Device was successful. The text below presents the contents of the CAN Data Field associated with the Forward Close Response returned to the Requesting Device:

**Table 10-6-4 Forward Close Response on DeviceNet**

Bytes	Meaning
07	Frag = 0, Transaction ID = 0, Destination MAC = 7.
DE	Service = DE. In this context, this specifies a response to the Forward Close service.
00	The General Status field. The value zero (0) indicates that there were no routing errors.
00	Reserved byte.
0C 00 FF FF 00 01 02 03 00 00	Forward Close Service Response Data. This is the Forward Close service data described above.



This page is intentionally left blank

## **Volume 3: DeviceNet Adaptation of CIP**

# **Appendix A: Explicit Messaging Services**

## **Contents**

A-1	Introduction.....	3
-----	-------------------	---

## **A-1 Introduction**

This chapter of the DeviceNet specification contains additions to the definition of CIP explicit messaging services that are DeviceNet specific. At this time, no such additions exist.

This page is intentionally left blank

## **Volume 3: DeviceNet Adaptation of CIP**

### **Appendix B: Status Codes**

---

## **Contents**

B-1	Introduction.....	3
-----	-------------------	---

## **B-1 Introduction**

This chapter of the DeviceNet specification contains additions to the definition of CIP error codes that are DeviceNet specific. At this time, no such additions exist.



This page is intentionally left blank

## **Volume 3: DeviceNet Adaptation of CIP**

# **Appendix C: Data Management**

---

## **Contents**

C-1	Introduction.....	3
-----	-------------------	---

## **C-1 Introduction**

This chapter of the DeviceNet specification contains additions to the definition of CIP Data Management that are DeviceNet specific. At this time, no such additions exist.

This page is intentionally left blank

## **Volume 3: DeviceNet Adaptation of CIP**

# **Appendix D: Engineering Units**

---

## **Contents**

D-1	Introduction.....	3
-----	-------------------	---

## **D-1 Introduction**

This chapter of the DeviceNet specification contains additions to the definition of CIP Engineering Units services that are DeviceNet specific. At this time, no such additions exist.



This page is intentionally left blank